Summer 2017

Exploiting Power for Smartphone Security and Privacy

Qing Yang
College of William and Mary - Arts & Sciences, qyang@email.wm.edu

Exploiting Power for Smartphone Security and Privacy

Qing Yang

Williamsburg, VA, USA

Master of Engineering, Chinese Academy of Sciences, China, 2007

A Dissertation presented to the Graduate Faculty
of The College of William & Mary in Candidacy for the Degree of
Doctor of Philosophy

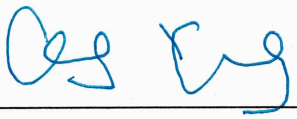
Department of Computer Science

College of William & Mary
January 2018

APPROVAL PAGE

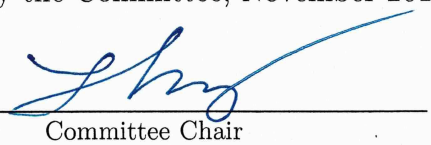
This Dissertation is submitted in partial fulfillment of
the requirements for the degree of

Doctor of Philosophy

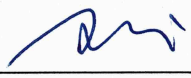


Qing Yang

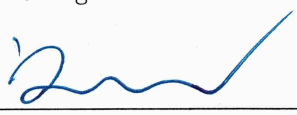
Approved by the Committee, November 2017



Committee Chair
Associate Professor Gang Zhou, Computer Science
College of William & Mary

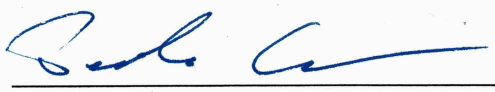


Professor Qun Li, Computer Science
College of William & Mary



Assistant Professor Xu Liu, Computer Science
College of William & Mary

Assistant Professor Bin Ren, Computer Science
College of William & Mary



Assistant Professor Paolo Gasti, Computer Science
New York Institute of Technology

COMPLIANCE PAGE

Research approved by

The College of William & Mary Protection of Human Subjects Committee

Protocol number(s): PHSC-2016-07-19-11319-gzhou

Date(s) of approval: 07/21/2016

ABSTRACT

Power consumption has become a key issue for smartphone security and privacy protection. In this dissertation, we propose to exploit power for smartphone security, as well as to optimize energy consumption for smartphone privacy.

First, we show that public USB charging stations pose a significant privacy risk to smartphone users. We present a side-channel attack that allows a charging station to identify which webpages are loaded while the smartphone is charging. To evaluate this side-channel, we collected power traces of Alexa top 50 websites on multiple smartphones under several conditions, including: varied battery charging level, browser cache enabled/disabled, taps/no taps on the screen, WiFi/LTE, TLS encryption enabled/disabled, different amounts of time elapsed between collection of training and testing data, and various hosting locations of the website being visited. The results of our evaluation show that the attack is highly successful: in many settings, we were able to achieve over 90% accuracy on webpage identification. On the other hand, our experiments also show that this side-channel is sensitive to some of the aforementioned conditions.

Second, we introduce a new attack that allows a malicious charging station to identify which website is being visited by a smartphone user via Tor network. Our attack solely depends on power measurements performed while the user is charging her smartphone. We evaluated the attack by training a machine learning model on power traces from 50 regular webpages and 50 Tor hidden services. We considered realistic constraints such as different Tor circuits types and battery charging levels. We were able to correctly identify webpages visited using the official mobile Tor browser with accuracy of up to 85.7% when the battery was fully charged, and up to 46% when the battery level was between 30% and 50%. Our results show that hidden services can be identified with higher accuracies than regular webpages.

Third, we propose a memory- and energy-efficient garbled circuit evaluation mechanism named MEG on smartphones. MEG utilizes batch data transmission and multi-threading to reduce memory and energy consumption. We implement MEG on Android smartphones and compare its performance with existing methods (non-pipelined and pipelined). Two garbled circuits of different scales, 128-bit AES encryption (AES-128) and 256-bit Levenshtein distance (EDT-256), are considered. Our measurement results show that compared with non-pipelined method, MEG decreases the memory consumption by up to 97.5% for EDT-256 when batch size is 2 MB. Compared with pipelined method, MEG reduces the energy consumption by up to 42% for AES-128 and 23% for EDT-256. Multi-thread MEG also significantly decreases the circuit evaluation time by up to 56.7% for AES-128 and up to 13.5% for EDT-256.

TABLE OF CONTENTS

ACKNOWLEDGMENTS

This dissertation is written with the support and help from many individuals. I would like to thank all of them.

First and foremost, I would like to express my deepest appreciation to my advisor, Dr. Gang Zhou. Without his guidance in my research, encouragement in my life, and confidence in my abilities, this dissertation would not have been possible.

I would also like to thank my dissertation committee, Dr. Qun Li, Dr. Xu Liu, Dr. Bin Ren, and Dr. Paolo Gasti, for serving on my Ph.D committee as well as their insightful comments.

My sincere thanks also go to all members of the LENS group past and present, Dr. Xin Qi, Dr. David T. Nguyen, Dr. Daniel Graham, Dr. Ge Peng, George Simmons, Kyle Wallace, Dr. Yantao Li, Dr. Shuangquan Wang, Amanda Watson, Hongyang Zhao, Yongsen Ma, and Woosub Jung, for the stimulating discussions, constructive suggestions, generous assistance, and effective teamwork.

Futhurmore, I would like to thank the faculty and staff at the Computer Science Department of the College of William & Mary. Special thanks to Vanessa Godwin, Jacqulyn Johnson, and Dale Hayes for their considerate and effective assistance.

Last but not the least, I would like to thank my family. Thanks to my parents, whose unwavering love and support has made me who I am today. Thanks to my wife, Yue Wang, for lighting up my life with so much love and joy.

This dissertation was supported in part by the U.S. National Science Foundation Secure and Trustworthy Cyberspace (SaTC) Grant #1618300 and CNS-1253506 (CAREER), and by a 2015 New York Institute of Technology Institutional Support for Research and Creativity (ISRC) Grant.

This dissertation is dedicated to my beloved parents, my lovely wife, and my brother for their endless and selfless love and support.

LIST OF TABLES

LIST OF FIGURES

Exploiting Power for Smartphone Security and Privacy

Chapter 1

Introduction

The extensive usage of smartphones brings increasing security and privacy risks for users. Because of the mobility of their devices, smartphone users are susceptible to attacks at many places, ranging from their homes to public areas. Smartphones also carry increasingly sensitive contents, such as web browsing history, communication messages, and financial transaction records. If such information leaks, the users will encounter privacy invasions and even financial losses. As communication and sensor techniques advance, smartphones feature many connection interfaces integrated, such as USB, WiFi, and NFC. Each interface is vulnerable to a breach caused by the exploitation of various design errors or software bugs. Furthermore, because smartphones have different characteristics from traditional computers, such as low CPU performance and limited battery capacity, existing successful security and privacy protection techniques for computers may not work on smartphones.

As users rely on smartphones for a variety of power-intensive activities, power consumption has become a key issue for smartphone security and privacy protection. This leads to two problems. The first problem is the risk of a power-based security attack. Because smartphones adopt aggressive dynamic power optimization techniques when running various apps, it is possible to infer user activity data based on power consumption traces. The second problem is the energy efficiency of privacy protection. Existing privacy protection techniques, such as secure computation, incorporate complex functions

that require huge computing capability and energy consumption. However, smartphone memory, CPU, and battery capacities are limited and have not kept up with the increase of complexity of privacy protection applications.

1.1 Problem Statements

In this dissertation, we propose how to exploit power for smartphone security, as well as how to optimize energy consumption for smartphone privacy. Specifically, we work on the following three problems.

(1) Inferring Browsing Activity on Smartphones via USB Power Analysis Side-channel Users commonly rely on their smartphones for a variety of energy-intensive activities such as video streaming, web browsing, connection sharing (hotspot), gaming, and VoIP. As a consequence, recharging smartphones multiple times a day has become common practice. A recent survey [1] found that users in the United States charge their smartphones anywhere from 1.8 to 2.6 times a day on average. To address users' charging needs, USB charging stations (or kiosks) are becoming ubiquitous in many public areas, including airports [2], parks [3, 4], hotels [5], and hospitals [6].

While charging stations undoubtedly provide convenience to smartphone users, they also expose them to privacy threats. For example, in an attack called "juice-jacking", an untrusted charging station exfiltrates data by covertly setting the smartphone into USB transfer mode [7]. To counter data exfiltration attacks, hardware devices such as SyncStop [8] are designed to physically interrupt data wires at the USB port, thereby blocking all data transfers. In this project, we show that even physically interrupting USB data wires does not prevent a charging station from extracting sensitive information from a smartphone. In particular, we demonstrate that a malicious charging station can infer smartphone browsing activity by analyzing USB power consumption patterns.

(2) USB Side-channel Attack on Tor. Tor [9] is an application-level low-latency anonymity network that enables anonymous communication between a client and arbitrary Internet destinations. Tor uses onion routers [10] hosted by volunteers to unlink

the identity and geographical location of the client from the server and to conceal the identity of the server from any adversary who can observe the client's network activity. Because of these properties, Tor is frequently used to provide anonymous connections to overcome communication restrictions and to evade censorship. Users rely on Tor to conceal their activities from governments, employers, and ISPs, since those might abuse, misuse, or accidentally leak sensitive information.

Because Tor is accessed by millions of users [11] including smartphone users (Tor Orbot, for instance, has more than 10 million installs on Android phones [12]), the security of Tor is becoming increasingly important. Given prior work on the security of widely-available public USB charging stations [2], in this work we investigate whether a malicious charging station can infer which websites are accessed by the Tor user while charging her smartphone. The ability to determine which website is being accessed through Tor using power consumption information, rather than through observation of network traffic, represents a hitherto unexplored and potentially devastating attack.

(3) Memory and Energy Efficient Garbled Circuit Evaluation on Smartphones. Secure computation involves multiple parties computing the value of a function without any outside parties. To protect the privacy of each party, no information about the input of any party is leaked to any other parties. Secure computation has wide usage in electronic voting, privacy-preserving data mining, and biometric-based authentication. Garbled circuit protocol is one solution for secure two-party computation. The basic idea of garbled circuit is that one party ("generator") transforms a function into a garbled boolean circuit and sends the circuit to another party ("evaluator") for circuit evaluation.

Currently, there are two types of circuit evaluation: no-pipelining and pipelining. The no-pipelining method stores the entire garbled circuit in the memory of the evaluator, a process which requires low power but high memory overhead. In the pipelining method, the garbled circuit generation and evaluation of different gates overlap so that lower memory overhead but higher power consumption, as compared to the no-pipelining

method, are needed. We propose the use of memory- and energy-efficient garbled circuit evaluation (MEG) on smartphones. The basic idea of MEG is to split the gates into batches and transmit the data in bursts. We investigate what burst size is best to minimize the energy consumption.

1.2 Contributions

This dissertation proposes three contributions towards exploiting power for smartphone security and privacy. The overall contributions are as follows.

Inferring Browsing Activity on Smartphones via USB Power Analysis Side-channel. We proposed a side-channel attack for a malicious charging station to infer smartphone browsing activity based on USB power consumption data. Specifically, we make two contributions.

- To fully characterize the side-channel associated with USB power consumption, we analyzed how webpage identification accuracy is impacted by variables pertinent to mobile devices, such as battery charging level, wireless connection type (WiFi or LTE), and the presence or absence of taps on the screen. To our knowledge, this is the first work that tests the feasibility of this attack against constraints commonly encountered in a mobile environment.

- We determined the impact of other variables that have not been considered in prior studies on power-based side-channels [13]. These include the effects of availability of the browser cache, training and testing signals collected on different smartphones, the time elapsed between the collection of training and testing signals, geographical proximity between the user and the web server, duration of power traces, and availability of encrypted (TLS) connections on identification accuracy.

USB Side-channel Attack on Tor. We proposed a new attack that exposes which website is being browsed by a smartphone user via Tor . Our main contributions are:

- We demonstrated a technique based on USB power analysis that allows a malicious charging station to identify which webpages are loaded on a smartphone using Tor. To our knowledge, this is the first work to study attacks on Tor based on smartphone power side-channels.

- We validated our attack under realistic smartphone constraints by collecting and analyzing power traces under several scenarios, including different networks (WiFi and LTE), different devices, and different battery charging levels. We correctly identified webpages visited using the official mobile Tor browser. We achieved accuracies between 36.58% and 85.7% when the battery was fully charged, and between 34.5% and 46% when the battery level was at 30%-50%. In comparison, accuracy obtained by random website selection is 1% for all websites, and 2% when considering hidden services or public webpages alone

Memory- and Energy- Efficient Garbled Circuit Evaluation on Smartphones. We introduce a new garbled circuit evaluation scheme on smartphones with improved memory and energy efficiency. Our main contributions are:

- We evaluated MEG using different burst size and compared it with existing circuit evaluation methods. The preliminary evaluation results show that the energy consumption of MEG is less than the current pipelining schemes and close to that of the no-pipelining method. MEG also significantly reduces the evaluation time as compared to basic pipelining.

- We propose two versions of MEG: single-threaded and multi-threaded. The multi-thread version is improved using a slow start technique to reduce the waiting time for the first bunch of data. We compared the energy consumption and time overhead of both versions.

1.3 Dissertation Organization

The rest of this dissertation is structured as follows. In Chapter 2, we discuss related work. In Chapter 3, we present our study of inferring browsing activity on smartphones via USB power analysis side-channel. In Chapter 4, we propose our method of USB side-channel attack on Tor network. In Chapter 5, we propose our system of memory and energy efficient garbled circuit evaluation on smartphones. Finally, we conclude the dissertation in Chapter 6.

Chapter 2

Related Work

This chapter reviews related work in power analysis side-channel attack, and passive and active attacks on Tor respectively.

2.1 Side-channel Attacks on Smartphones

We review prior work on: (1) side-channel attacks using power analysis; (2) webpage fingerprinting; and (3) attacks on smartphones via USB port.

Side-channel Attacks Using Power Analysis Clark et al. [13] identified webpages loaded on a computer by measuring power consumption at the AC wall outlet. Power traces were analyzed in the frequency domain, and matched using a SVM classifier. The evaluation results showed that this side-channel attack achieved 86.75% precision and 74% recall.

The main differences between our work and [13] are as follows. (1) Ours is the first to study this side-channel on smartphones, which drastically differ in architecture from desktops and laptops, and have hardware (e.g., a touchscreen) not usually associated with traditional computers. Additionally, smartphones adopt aggressive dynamic power optimization techniques [14] that could interfere with the side-channel. (2) In contrast with [13], our work explores how the identification accuracy is affected by variables such as battery charging level, user's interaction with the touchscreen, trace length, time

between training of our identification model and testing, type of wireless connection (WiFi and LTE), and website characteristics (HTTP vs HTTPS, geographical location). (3) The results reported in [13] were obtained with 15-second traces, compared to 2- to 6-second traces in our work. We consider this an important distinction, since trace length has a large impact on identification accuracy, and most users spend *less* than 15 seconds on average on a webpage [15].

Several papers analyzed power data from sources other than USB charging ports or AC outlets to extract cryptographic keys or other private information. For instance, Genkin et al. [16] demonstrated a side-channel attack based on electric potential from computer chassis to extract RSA and ElGamal keys. Their work is based on the observation that the fluctuation in electric potential of the chassis correlates with computation.

Michalevsky et al. [17] introduced PowerSpy, a tool that analyzes aggregate power consumption on the phone during a period of several minutes to infer the user's location. The authors observed that the power consumption of a smartphone is measurably affected by the smartphone's distance from the surrounding cellular base station. To obtain instantaneous power consumption, PowerSpy uses unprivileged Android APIs. For this reason, PowerSpy infers location information without requesting any explicit permission to the user, in contrast with the use of *privileged* location APIs—which require explicit user confirmation.

Lin et al. [18] demonstrated that it is possible to identify running apps, details about on-screen content, password lengths, and geographical locations by measuring instantaneous power consumption. Our work differs from [18] in the following ways. (1) Li et al. do not focus on website identification. (2) They use a different adversary model, which assumes that the adversary is able to install a malicious app on the smartphone, or is able to measure power consumption directly at the battery connectors inside the smartphone, rather than at the USB port. Our attack uses USB charging port to obtain power traces, and does not require the smartphone to run malicious software. (3) Li et al. do not consider factors common to smartphones, such as the current battery level

or the type of network used to retrieve data. These factors, as demonstrated by our experiments, affect the accuracy of the power-based side channel attack.

Webpage Fingerprinting via Traffic Analysis Several side-channel attacks use network traffic analysis to infer user's web browsing activities. For example, Hintz et al. [19] demonstrated that transferred file sizes could be used as a reliable fingerprint for webpages. Similarly, Lu et al. [20] exploited both packet size and packet ordering information to improve webpage identification success rate. Additionally, studies have shown that encrypted channels do not protect against traffic analysis. For example, Chen et al. [21] were able to infer browsing activity via packet analysis on traffic encrypted using HTTPS and WPA. Our work adds to the current body of work on webpage fingerprinting by demonstrating a new and complementary side-channel.

USB Data Port Vulnerabilities Karsten et al. [22] and Andy et al. [23] demonstrated that it is possible to install malicious software on smartphones via their USB port by exploiting vulnerabilities in mobile operating systems. To prevent these attacks, the authors suggest to disable/remove data pins in the USB cable. Unfortunately, our work shows that even if data pins are removed, the charging station can still learn information about the user's browsing activity.

2.2 Attacks on Tor Network

There are a number of papers focusing on attacks on Tor. These papers can be broadly categorized into *passive* attacks (i.e., based on traffic analysis) and *active* attacks (based on traffic modification).

Passive Attacks Based on Traffic Analysis Fingerprinting attacks and traffic confirmation attacks belong to this category. Website fingerprinting attacks enable an attacker to detect patterns that are indicative for webpages in Tor traffic. Herrmann et al. [24] presented a method that applies common text mining techniques to the normalized frequency distribution of observable IP packet sizes, so as to reveal requested websites.

Panchenko et al. [25] showed that Tor did not offer sufficient security against web site fingerprinting. Their analysis relies on volume, time, and direction of the traffic to fingerprint websites. Cai et al. [26] presented a web page fingerprinting attack that was able to defeat several recently proposed defenses against traffic analysis attacks, including application-level defenses HTTPOS and randomized pipelining over Tor. Abbott et al. [27] provided an attack to identify a fraction of the Tor users who used a malicious exit node. This attack tricked a user's web browser into sending a distinctive signal over the Tor network. Such signal could be detected by traffic analysis.

In traffic confirmation attacks, the adversary must be able to eavesdrop both ends of a communication over a long period of time. Levine et al. [28] investigated timing analysis attacks on low-latency mix systems, and proposed a novel technique, called defensive dropping, to mitigate timing attacks. Hopper et al. [29] presented two attacks on low-latency anonymity schemes using the network latency information. The first attack allowed a pair of colluding web sites to predict, whether two connections from the same Tor exit node are using the same circuit with high confidence. The second attack allowed a malicious web site to gain several bits of location information about a client each time he visits the site. Sun et al. [30] presented asymmetric traffic correlation attack on Tor network with 95% accuracy, and increased the threat of AS-level attacks by 50% to 100%. Bauer et al. [31] demonstrated that Tor routing optimizations impact its ability to provide strong anonymity. They further proposed attacks using low-resource Tor nodes to compromise the entrance and exit servers on a Tor path. Kwon et al. [32] presented the first practical passive attack against hidden services and their users, based on circuit fingerprinting attack. With this attack, the adversary can identify the presence of (client or server) hidden service activity in the network with high accuracy. Murdoch et al. [33] introduced new traffic-analysis techniques based on a partial view of the network. Their attack could infer which nodes were used to relay the anonymous streams and therefore greatly reduced the anonymity provided by Tor. Chakravarty et al. [34] presented a remotely-mounted attack to expose the network identity of an anonymous client, hidden

service, or anonymizing proxy. They employed single-end bandwidth estimation tools and a colluding network entity to modulate traffic directed to the victim.

Our work differs from these studies mainly in the following aspects: (1) the attack presented in this project is based on USB power analysis, rather than network traffic analysis or modification; (2) we focus on web page identification for both regular web pages and web pages served by Tor hidden services; and (3) we study a side-channel attack on smartphones, rather than on desktop or laptop computers.

Active Attacks Against Tor Network Wang et al. [35] investigated the fundamental limitations of flow transformations in achieving anonymity. They showed that flow transformations could not necessarily provide the level of anonymity people expected or believed. Barbera et al. [36] introduced a new Denial-of-Service attack against Tor Onion Routers. They exploited a design flaw used by Tor software to build virtual circuits. Their attack only needed a fraction of the resources required by a network DoS attack to achieve similar damage on the Tor network.

Our work differs from the above papers because it does not require active changes to the content of webpages, or traffic injection or manipulation.

There were also efforts on side-channel attack via USB power analysis. Yang et al. [37] presented a side-channel attack on mobile devices that allows an adversary to identify loaded webpages while the smartphone is charging by controlling the USB charging port. Our attack is different from previous work in that: (1) all webpages are loaded using Tor, which has significant impacts on the loading process, (2) besides public webpages, we also consider hidden services that are exclusively existent in Tor network, and (3) the impacts of factors unique for Tor, such as circuit type, are evaluated. Spolaor et al. [38] demonstrated that sensitive data could be sent as power bursts from a smartphone to a malicious charging station over a USB cable only with power pins. Their attack needed to install an app on the phone to modulate the data first. In contrast, our technique solely depends on measured power consumption to infer the data, not requiring any additional apps.

2.3 Optimization on Garbled Circuit

There are several papers on garbled circuit optimization. These works can be categorized into general optimization and smartphone-focused optimization.

General Optimization of Garbled Circuit Protocol Goyal et al. [39] proposed a method where the generator uses random seed to construct the circuit and later only sends the seeds (instead of of the whole circuit) to the evaluator. This technique helps to save the communication traffic of circuit evaluation. Kolesnikov et al. [40] proposed a new garbled circuit construction method, where a garbled XOR gates can be replaced by a standard XOR operation and evaluated "for free". This technique helps to reduce the computation and communication cost for circuit evaluation. Pinkas et al. [41] presented several algorithmic improvements for garbled circuit protocol. They evaluated these optimizations using reasonably large circuits, such as AES encryption. Huang et al. [42] presented several techniques, such as fast table lookup, for improving the running time and memory requirements of the garbled-circuit technique, resulting in an implementation of generic secure two-party computation that is significantly faster than previously reported while also scaling to arbitrarily large circuits. [43] assessed the feasibility of two-party secure computation in the presence of a malicious adversary. They showed that evaluating billion-gate circuits is feasible when using the existing techniques and parallelizing steps in the protocol. Lindell et al. [44] presented a protocol using cut-and-choose to boost to be secure for malicious adversaries. Their protocol is more efficient and simpler than protocols with the same methodology.

Garbled Circuit Optimization for Smartphones Huang et al. [45] presented a Java-based secure computation framework for Android smartphones. They found that processing power, instead of the bandwidth, is the biggest performance obstacle for secure computation on smartphones. Mood et al. [46] developed a new methodology of generating garbled circuits that is memory-efficient. They used the standard SFDL language for describing secure functions as input, and designed a new pseudo-assembly

language and a template-driven compiler to generates circuit. The evaluation results on Android devices showed up to 95.6% memory overhead reduction for the 2-set case.

Sedenka et al. [47] designed privacy-preserving protocols for scaled Manhattan and scaled Euclidean verifiers, which are secure against malicious clients and honest-but-curious server. They augmented their protocols with principal component analysis that helps to improve authentication accuracy. Carter et al. [48] created a new SFE protocol that allows mobile devices to securely outsource the majority of computation required to evaluate a garbled circuit. Their protocol contained a new out-sourced oblivious transfer primitive that requires significantly less bandwidth and computation than standard OT primitives, as well as an outsourced input validation techniques that force the cloud to prove that it is executing all protocols correctly. Gasti et al. [49] proposed a privacy-preserving protocols for continuous authentication that securely outsources computation to an untrusted cloud. Their technique significantly decreases the energy requirement and running time for computing Manhattan distance and Hamming distance.

Our work differs from the above papers because our method is based on splitting gates into bursts, and we evaluate the effects of burst sizes. We also improve the current implementation using multi-threads.

Chapter 3

On Inferring Browsing Activity on Smartphones via USB Power Analysis Side-channel

3.1 Introduction

Users commonly rely on their smartphones for a variety of energy-intensive activities such as video streaming, web browsing, connection sharing (hotspot), gaming, and VoIP. As a consequence, recharging smartphones multiple times a day has become common practice. A recent survey [1] found that users in the United States charge their smartphones anywhere from 1.8 to 2.6 times a day on average. To address users' charging needs, USB charging stations (or kiosks) are becoming ubiquitous in many public areas, including airports [2], parks [3, 4], hotels [5], and hospitals [6].

While charging stations undoubtedly provide convenience to smartphone users, they also expose them to privacy threats. For example, in an attack called "juice-jacking", an untrusted charging station exfiltrates data by covertly setting the smartphone into

USB transfer mode [7]. To counter data exfiltration attacks, hardware devices such as SyncStop [8] are designed to physically interrupt data wires at the USB port, thereby blocking all data transfers. In this work, we show that even physically interrupting USB data wires does not prevent a charging station from extracting sensitive information from the smartphone. In particular, we demonstrate that a malicious charging station can infer smartphone browsing activity by analyzing USB power consumption patterns. In our experiments, we were able to identify with high accuracy which webpage the user was visiting, by applying standard machine learning techniques to power traces.

We believe that our attack represents a serious threat to privacy, because: (1) USB charging stations are becoming widespread around the world, and therefore more and more users can be targeted by this attack; and (2) the modifications required to implement the attack can be easily concealed (e.g., see [50]), and can therefore go unnoticed by users.

3.1.1 Contributions and Findings

To fully characterize the side-channel associated with USB power consumption, we analyzed how webpage identification accuracy is impacted by variables pertinent to mobile devices, such as battery charging level, wireless connection (WiFi or LTE), and taps on the screen. To our knowledge, this is the first work that tests the feasibility of this attack against constraints commonly encountered in a mobile environment. In addition, we determined the impact of other variables that have not been considered in prior studies on power-based side-channels [13]. These include availability of browser cache, training and testing signals collected on different smartphones, the time elapsed between the collection of training and testing signals, geographical proximity between the user and the web server, duration of power traces, and availability of encrypted (TLS) connections on identification accuracy. Next, we summarize our findings.

Impact of Battery Charge Level and Taps on Screen Both charging the battery and tapping on the screen reduced webpage identification accuracy. Identification ac-

curacy decreased when the smartphone's battery was charging at 30% level, compared to when the battery was fully charged. However, even with the decrease in accuracy, it was still possible to reliably infer browsing information. Similarly, taps on the screen added significant noise to the power traces, making webpage identification challenging. Our results show that this factor caused a significant degradation in identification performance.

Impact of Other Variables The time elapsed between collection of training and testing traces had a major impact on webpage identification accuracy, with training traces older than 30 days leading to a significant drop in identification accuracy. This suggests that traces used to train the classifiers should be updated frequently to improve the attack's success rate. Increasing the duration of power traces for training and testing led to improved identification accuracy. We were able to reliably identify webpages under both WiFi and LTE, even when the training power traces were collected using one type of connectivity (e.g., LTE), and the testing traces were obtained with another (e.g., WiFi). When using different smartphones for training and testing, accuracies dropped significantly. However, using two smartphones for training, and a different smartphone for testing reduced the drop in webpage identification accuracies.

When the user did not tap on the screen, enabling browser cache improved identification accuracy. However, for power traces collected when the user tapped on the screen and while the smartphone was charging, enabling cache led to a decrease in webpage identification accuracies.

Our results show that increasing the geographical distance between the smartphone and the host serving the webpages reduced identification accuracy. When we divided webpages in foreign (located outside of the continental United States), and local (within the United States), we observed that webpages hosted locally had slightly higher identification accuracies than foreign-hosted webpages.

Finally, retrieving webpages via secure connections (HTTPS or, more specifically, TLS) did not have a measurable impact on identification accuracy.

Experiment Results In our experiments, we used machine learning algorithms to identify which webpage the user visited out of a closed set of fifty webpages [51]. We were able to achieve identification accuracies as high as 98.8% with 2-second traces. Even in the worst case, i.e., when the cache was enabled, the user tapped on the screen, and the battery was charging from 30%, we achieved an identification accuracy of 54.2% with 6-second traces. When training and testing traces were collected using different smartphones, identification accuracy was at least 44.5%. This is significantly higher than choosing one out of fifty webpages at random (which leads to 2% baseline accuracy).

3.2 Experiment Setup

We collected power traces while webpages were loading on two types of smartphones: Samsung Galaxy S4, and Samsung Galaxy S6. We used the homepages of the 50 most popular (non-adult) websites, based on Alexa ranking [51] (see Table 3.1). These websites represent over 30% of the page views on the Internet [13]. To collect power traces during webpage loading, we instrumented the USB charging circuit as shown in Figure 3.1. Our circuit connects a DC power supply, a smartphone, and a data acquisition card (DAQ), and measures voltage variations (and therefore the corresponding power consumed) across a 0.1 Ω shunt resistor. To satisfy the USB charging specifications [52], we connected the data pins of the USB cable using a 200 Ω resistor.

Table 3.1: Alexa top 50 non-adult websites, as of June 2015. The table reports whether each website is foreign or domestic, and weather and the connection type is HTTP or HTTPS.

Webpage	Domestic	HTTPS	Webpage	Domestic	HTTPS	Webpage	Domestic	HTTPS	Webpage	Domestic	HTTPS
google.com	✓	✓	sina.cn			vimeo.com	✓	✓	fc2.com	✓	
facebook.com	✓	✓	weibo.com			imgur.com	✓		snapdeal.com		
youtube.com	✓		tmall.com			wordpress.com	✓	✓	ask.com	✓	
yahoo.com	✓	✓	ok.ru			cnet.com	✓		alibaba.com	✓	
baidu.com			ebay.com	✓		msn.com	✓		espncricinfo.com	✓	
wikipedia.com	✓	✓	about.com	✓		pinterest.com	✓	✓	360.cn		
amazon.com	✓		hao123.com			yandex.ru			stackoverflow.com	✓	
twitter.com	✓	✓	reddit.com	✓		paypal.com	✓	✓	netflix.com	✓	✓
taobao.com			bing.com	✓		microsoft.com	✓		163.com		
live.com	✓		etsy.com	✓	✓	vk.com			slideshare.net	✓	
qq.com			instagram.com	✓	✓	aliexpress.com	✓		craigslist.org	✓	
bankofamerica.com	✓	✓	sohu.com			apple.com	✓				
linkedin.com	✓	✓	tumblr.com	✓	✓	imdb.com	✓				

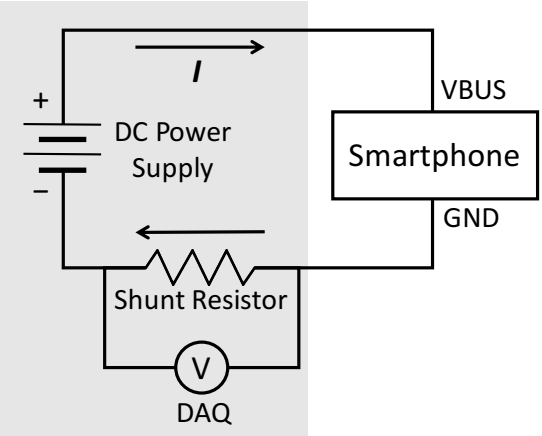


Figure 3.1: Overview of the setup used to collect power traces.

Most smartphones use lithium-ion (Li-ion) batteries due to their high energy density. The charging profile of Li-ion batteries encompasses two stages [53]. In the first stage, the smartphone charging circuit applies *constant current* to the battery. This stage ends when the battery reaches a specific charging voltage (usually between 3.7 V and 4.2 V). In the second stage, the battery is charged at a *constant voltage*, and the current gradually decreases until it reaches a termination value. Because the battery charging process could take several hours, the current used to charge the battery does not vary significantly while the smartphone is loading a webpage.

We used an Agilent E3630A DC power supply [54] as the power source. We measured the voltage drop across the shunt resistor using a National Instrument USB-6211 (DAQ) [55] at a sampling rate of 200 kHz. We set the power supply to output a fixed voltage of 5.5 V. This voltage is higher than the nominal USB voltage of 5 V to compensate for the voltage drop introduced by the shunt resistor. The resulting voltage was between 5.32 V and 5.48 V, which is within the tolerance of many modern smartphones [52]. The DAQ's data output was connected to a laptop, which stored data for offline analysis using LabVIEW. Figure 3.2 shows the power consumption traces collected while loading the homepages of `google.com` and `youtube.com`.

We collected power traces in two modes: *user-actuated*, and *automated*. With user-actuated traces, the user initiates webpage loading by typing a URL in Mobile Chrome's

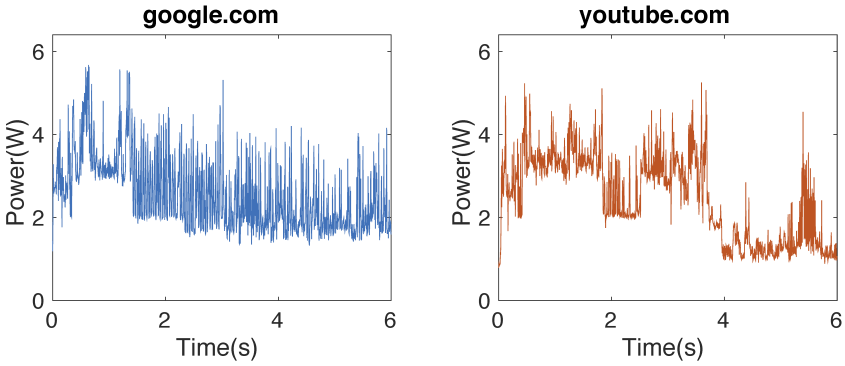


Figure 3.2: Power traces collected during the first 6 seconds of automated webpage loading activity. The left and right panels show power traces collected while loading `google.com`, and `youtube.com`, respectively. The x axis shows time (in seconds) from the beginning of the webpage loading activity, and the y axis shows the power consumed by the smartphone.

address bar. To collect automated traces, we developed an Android application that launches the Chrome browser, and uses it to load the intended webpage. Our application allows 10 seconds for webpage loading (only the first 6 seconds of data were recorded), and then loads the next webpage. Before each measurement, we closed all other applications on the smartphone, and set the screen brightness to a constant level.

User-actuated and automated traces were collected under two conditions: battery level (30% vs. 100%).[1] and browser cache (enabled vs. disabled). We chose these conditions because they impact smartphone energy consumption. When the battery is fully charged, almost all power from the charger is used to load webpages. In contrast, when the smartphone is charging, a sizable (almost constant) amount of power is used to charge the battery, hence affecting the traces. Cache availability was chosen because cache misses increase network activity, and therefore radio activity. Retrieving data wirelessly requires more energy than loading it from local flash memory.

We collected 40 automated traces per webpage for each of the following combinations: 30% battery, cache; 30% battery, no cache; 100% battery, cache; 100% battery, no cache.

[1]We implemented the 30% battery level condition as follows. We charged the smartphone battery to 30% level, and then started collecting power traces while still charging the smartphone. As the battery reached 35% level, we interrupted data collection, discharged it to 30%, and resumed the collection of power traces.

Additionally, we collected 10 user-actuated traces for the same combinations. To collect traces, we used four Samsung Galaxy S4 devices (in the rest of this work, we refer to these devices as D1, D2, D3, and D4) in most of the experiments. To analyze the impact of different smartphone models on the attack, we also conducted experiments on a Samsung Galaxy S6 (referred to as D5). We collected all traces in Old Westbury (NY), and in Williamsburg (VA). Further details on our datasets and their usage in our work are provided in Table 3.2.

Table 3.2: Details on the dataset used in this work. The devices used for data collection are four Samsung Galaxy S4 (labelled D1, D2, D3, and D4 in the table), and one Samsung Galaxy S6 (D5).

Location	Device(s)	Network	Collection Method	Browser Cache	Battery Level	Collection Time	Usage in our work
Old Westbury (NY)	Galaxy S4 (D1)	WiFi	Automated	Disabled	100%	May 2015	Table 3.8
						June 2015	
						July 2015	Table 3.3, 3.8, 3.9, 3.10
					30%	July 2015	Table 3.3, 3.9, 3.10
				Enabled	100%	Aug. 2015	Table 3.3, 3.9, 3.10
					30%		
			User-actuated	Disabled	100%	July 2015	Table 3.4
					30%		
				Enabled	100%	Aug. 2015	
					30%		
Williamsburg (VA)	Galaxy S4 (D2)	WiFi	Automated	Disabled	100%	Aug. 2016	Table 3.5
		LTE					
	Galaxy S4 (D2, D3, and D4)	WiFi	Automated	Disabled	100%	Aug. 2016	Table 3.6
					30%	Sept. 2016	Table 3.7
	Galaxy S6 (D5)	WiFi	Automated	Disabled	100%	Aug. 2016	Table 3.3
					30%	Sept. 2016	

3.3 Webpage Identification

Our webpage identification process consists of *training* and *testing* phases. In the training phase: (1) we extracted frequency-domain features from the power traces; and (2) we trained a classifier (Random Forest [56]) on the extracted feature vectors. During the testing phase, we use the trained classifier to predict webpage labels on new data. To improve identification accuracy, we performed feature extraction on partially overlapping segments from traces, and implemented classifier voting. Next, we provide details on feature extraction, classification, and trace segmentation.

Classifier Training and Testing We used Random Forest [56] to classify power traces

because in our experiments it outperformed other commonly used classifiers, such as SVM [57, 58], and Dynamic Time Warping (DTW) [59]. We used the WEKA [60] implementation of Random Forest.

We experimented with four training-testing scenarios. The first involved 40 power traces per webpage, collected using automated webpage loading; 20 traces were used for training the classifier, and the remaining 20 traces for testing. This scenario is used when training and testing are performed with data from the same smartphone, such as in tables 3.3, 3.5, 3.8, 3.9, 3.10.

In the second scenario, we trained the classifier using all 40 automatically-collected traces, and performed testing with 10 traces collected via user-actuated page loading. This scenario was used with data is collected with user taps, i.e., in Table 3.4.

In the third scenario, we trained our classifier using 40 traces per webpage, collected using automated webpage loading, on one smartphone device; we then used 40 traces collected from a different smartphone device for testing. This scenario was used in tables 3.6 and 3.7, and in Figure 3.6.

Finally, in the fourth scenario, we trained the classifier using 80 traces from two smartphones (40 traces from each device), and tested on 40 traces from a different smartphone. This scenario was used in tables 3.6 and 3.7, and in Figure 3.6.

Feature Extraction We transformed each power trace to its corresponding frequency-domain representation using Fast Fourier Transform (FFT) [61]. To reduce the impact of noise on individual frequencies, we divided the frequency range into equal-size bins. For each bin, we calculated the average amplitude of all the frequencies in it. The average amplitudes were used as features. We experimented with different numbers of bins, and settled on using 125 bins (resulting in 125 features) when using the same device for training and testing, because this value led to the highest identification accuracy. When using different devices and smartphone models during training and testing, we used the first 15 of 125 bins (i.e., first 15 features) corresponding to the 15 lowest frequencies, which allowed us to achieve the highest identification accuracy for this setting. Figure 3.3

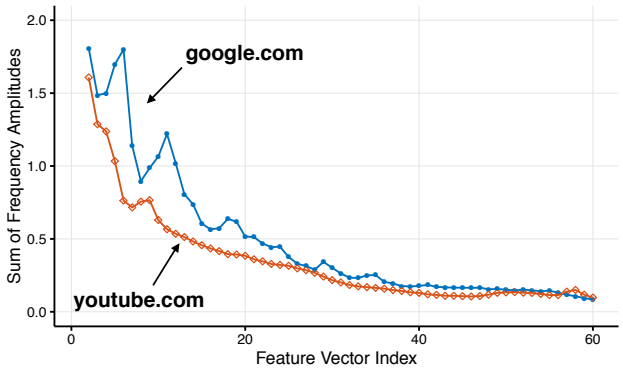


Figure 3.3: Aggregate amplitudes corresponding to the first 60 bins computed from the power traces of `google.com` and `youtube.com`. (Corresponding time-domain traces are illustrated in Figure 3.2.) The x axis represents the index of each bin.

shows the result of feature extraction on the data in Figure 3.2. Each data point in Figure 3.3 represents a feature.

Trace Segmentation and Voting Variable network conditions, web-server load, and smartphone background applications introduce intermittent noise in power traces. To mitigate the effects of noise, we divided each trace into overlapping 0.5-second segments. Feature extraction was performed on each segment, and the classifier was trained using segments from all traces. For testing, we classified individual segments of a session, and used the classification results as votes. Each trace was then assigned to the class that received the largest number of votes. This strategy of segmenting and using majority voting led to an improvement in accuracy between 0.1% and 7.7% compared to using entire traces.

Evaluation of Identification Performance To evaluate classifier performance, we calculated Rank 1 and Rank 5 identification accuracies. With Rank 1, a trace is classified correctly if the most popular label assigned to the trace's segments is the correct label for the trace. In case of ties, the classifier outputs the label with the highest associated confidence, defined as Random forest leaf node probability. With Rank 5, we consider a trace as correctly classified if the correct label appears within the 5 most popular labels.

Because our identification task involves predicting 50 webpages, the baseline accura-

cies, obtained by randomly guessing the webpages, are 2% for Rank 1 and 10% for Rank 5.

Finally, for each rank, we also present the Normalized Rank-n Accuracy, which is defined as follows. Let p_n be the probability that the classifier correctly labels a trace for Rank-n. The probability of correctly guessing the website loaded by the smartphone is computed as p_n/n, and represents the probability that the adversary guesses the correct website label given the Rank-n output of the classifier.

We consider this metric because it represents the uncertainty an adversary encounters in identifying the website correctly as the rank increases. For example, if Rank 2 accuracy is 60%, and Rank 5 accuracy is 95%, then the normalized Rank 2 accuracy is 30%, while the normalized Rank 5 accuracy is 19%. This shows that if the adversary is interested in identifying a unique website visited by the user, Rank 2 leads to better results. On the other hand, if the adversary only needs to know if the user is visiting any of the five webpages identified in Rank 5 (e.g., because they are all from social networking websites), then the Rank 5 result is more meaningful. Therefore, the information provided by this metric complements the Rank 1 and Rank 5 accuracy results presented throughout the project.

3.4 Evaluation

We evaluated identification accuracy under three variables: (1) trace length (2 s, 4 s, and 6 s), (2) cache enabled vs. disabled, and (3) battery charge level (30% vs. 100%). In this section, we report how these variables impact identification accuracy for *automated* and *user-actuated* data collection. Identification accuracies under other variables, such as different smartphones used in training and testing, type of wireless connectivity, time elapsed between collection of training and testing traces, location of the hosts serving the webpages, and HTTP vs. HTTPS, are reported in sections 3.5 and 3.5.

Table 3.3: Identification Accuracy (in %) using frequency-domain features and classifier voting for automated dataset collected using D1. For comparison, results from a Samsung Galaxy S6 (D5) are also reported. All experiments were performed using 125 features.

	Rank 1			Rank 5		
	2 s	4 s	6 s	2 s	4 s	6 s
Charged, no cache	96.7	99.0	99.3	99.0	99.4	99.8
Charged, w/ cache	98.8	99.6	100	99.6	100	100
Charging (30%), no cache	82.8	91.7	95.7	92.8	96.7	98.5
Charging (30%), w/ cache	87.6	94.4	96.2	94.5	97.4	97.6
S6, charged, no cache	84.3	94.8	97.1	94.5	98.6	99.5
S6, charging (30%), no cache	75.2	82.8	90.5	91.8	93.7	97.8

3.4.1 Identification Accuracy on Automated Dataset

Our results for the automated dataset are reported in Table 3.3. We achieved identification accuracy of at least 82.8% for Rank 1, and at least 92.8% for Rank 5 using a Samsung Galaxy S4, with 2 second traces. With a Samsung Galaxy S6, accuracies were at least 75.2% for Rank 1, and at least 91.8% for Rank 5. Next, we discuss how each variable affects identification accuracy.

Trace Duration Increasing the duration of the traces led to an improvement of identification accuracy. Although we achieved the highest identification accuracy with six-second traces, we also had good identification accuracies (Rank 1: 82.8%-98.8%) with 2-second traces, as shown in Table 3.3. Given that most webpages load within a few seconds on smartphones (2.9 s to 5 s on average, according to [62]), our results indicate that the attack correctly identifies a webpage within the typical loading time.

Caching Our results show that enabling cache improved identification accuracy (see Table 3.3). This is further validated by our results, reported in Table 3.9, where enabling cache improved identification accuracy for foreign-hosted websites more than for websites located within the United States. This is because the farther the host serving the content, the more network-related noise is added to the traces (see Section 3.5 for further details).

Battery Level Users connect their smartphones to charging ports at various battery levels [1]. Our experiments show that this variability impacts identification accuracy. In particular, we were consistently able to classify traces with higher accuracy when the

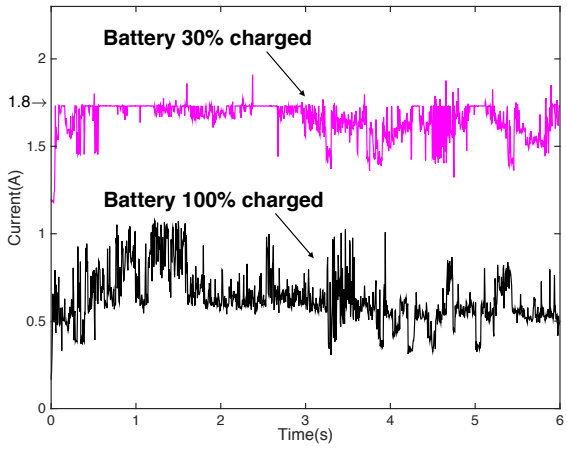


Figure 3.4: Power traces obtained while loading the same webpage (`yahoo.com`) with 30% and 100% battery level. The figure shows that the trace corresponding to 100% battery level exhibits a relatively higher dynamic range, because it is not capped by the 1.8 A limit which is often reached by the trace corresponding to 30% battery level.

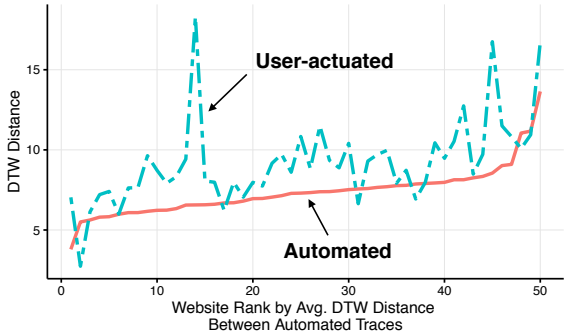


Figure 3.5: Average inter-class DTW distance on automated traces for each webpage. Measurements are performed with battery fully charged, and with cache enabled. Data is sorted in ascending order of average DTW distance.

battery was fully charged (see Table 3.3).

This can be explained by comparing the power traces illustrated in Figure 3.4. The USB charging specifications set an upper bound of 1.8 A to the amount of current that the smartphone can draw from the USB port. If the phone is charging, a substantial amount of the available current is directed to the battery, and therefore the fluctuations in power consumption due to webpage loading is limited. This is evident in Figure 3.4, where the signal corresponding to 30% battery has considerably lower variability compared to the signal collected with fully charged battery.

3.4.2 Identification Accuracy on User-Actuated Dataset

Once we included user activity in the form of taps, identification accuracy dropped significantly due to tap-induced noise. This is because tap characteristics (e.g., tap location on the screen, timing, and duration) are different in each trace, which leads to noisy traces. To validate this observation, we computed the average (intra-class) Dynamic Time Warp (DTW) distance between pairs of user-actuated traces, and between pairs of automated traces, under different caching and charging conditions. The average distance between user-actuated traces was consistently higher than the distance between automated traces. With cache enabled and fully charged battery, the average DTW distance between user-actuated traces was 17%-38% higher than the average distance between automatically-collected traces. When the smartphone was charging from 30% level, and with cache enabled, the DTW distance increased by 34%-86% when including taps. Once cache was disabled, the average DTW distance increased by 5%-15% with taps and with the battery fully charged. With no cache and battery charging from 30% level, the average DTW distance between automated and user-actuated traces increased by 31%-56%. Figure 3.5 shows the relative distance for each webpage with fully charged battery and cache enabled.

While two seconds were sufficient to classify webpages with high confidence using automated traces, this was not the case with traces from the user-actuated dataset. Regardless of caching and charging, we achieved good Rank 1 accuracy with six-second traces. In this setting, we correctly identified webpages between 54.2% and 88.4% of the times (for reference, selecting a webpage at random out of 50 leads to 2% accuracy). We achieved good Rank 5 accuracy (53.6% to 95%, compared to 10% with random chance) with four- and six-second traces. Charging had still a measurable impact on identification accuracy: identification on a fully charged phone consistently led to better results, all other variables being the same. Enabling cache with fully charged battery led to a 2.2%-5.8% improvement in identification accuracy. However, when we enabled cache in traces collected at 30% battery level, webpage identification accuracy dropped significantly. We

Table 3.4: Identification accuracy (in %) using frequency domain features and classifier voting for user-actuated traces (i.e., with taps) collected using D1

. All results were obtained using 125 features.

	Rank 1			Rank 5		
	2 s	4 s	6 s	2 s	4 s	6 s
Charged, no cache	9.0	56.0	82.6	19.0	79.2	92.8
Charged, w/ cache	7.0	58.6	88.4	22.4	83.0	95.0
Charging (30%), no cache	5.8	39.2	70.0	18.8	70.0	88.4
Charging (30%), w/ cache	4.2	35.8	54.2	15.8	53.6	72.0

Table 3.5: Identification Accuracy (in %) using frequency-domain features and classifier voting for automated dataset collected using D2. All experiments were performed with a fully-charged battery, and no cache, using 125 features.

	Rank 1			Rank 5		
	2 s	4 s	6 s	2 s	4 s	6 s
LTE training, LTE testing	82.7	98.2	99.7	94.2	100	100
LTE training, WiFi testing	24.9	55.2	73.9	45.2	72.8	86.0
WiFi training, LTE testing	23.8	68.6	84.8	50.3	85.1	95.7

hypothesize that this happened because taps on screen and battery charging contribute to a substantial increase in noise in power traces. Under these conditions, networking becomes an important source of discriminability between website traces. Enabling cache reduces the contribution of the network component to the power traces, in turn reducing classification accuracy.

Overall, our experiments show that although the presence of taps substantially reduces identification accuracy compared to automated collection of power traces, it is still possible to accurately classify six-second user-actuated traces.

3.5 Impact of Other Variables

We examined the identification accuracies according to the following variables: (1) different smartphones used for training and testing (training traces were collected from one or more smartphones that are not used for testing); (2) LTE and WiFi training and testing; (3) aging of training traces; (4) domestic vs. foreign websites, and (5) websites accessible via unencrypted connections (denoted as "HTTP") vs. accessible through TLS-encrypted links (denoted as "HTTPS"). Table 3.1 in Appendix indicates which websites are local,

Table 3.6: Identification Accuracy (in %) using frequency-domain features and classifier voting for automated dataset. All experiments were performed with a fully-charged battery, and no cache, using 15 features.

Training \ Testing	D3			D4			D2		
	2 s	4 s	6 s	2 s	4 s	6 s	2 s	4 s	6 s
D3	64.3	89.9	94.9	48.8	73.0	77.5	34.3	60.0	69.7
D4	48.4	73.1	79.9	67.7	93.5	96.6	43.1	71.1	75.0
D2	34.3	63.0	69.8	44.1	74.2	75.9	66.0	91.1	94.6
D4+D2	52.7	78.4	84.3						
D3+D2				56.7	84.4	85.8			
D3+D4							45.9	75.9	81.1

Table 3.7: Identification Accuracy (in %) using frequency-domain features and classifier voting for automated dataset. All experiments were performed with the battery charging from 30%, and no cache, using 15 features.

Training \ Testing	D3			D4			D2		
	2 s	4 s	6 s	2 s	4 s	6 s	2 s	4 s	6 s
D3	48.5	67.8	79.6	40.3	56.6	59.9	31.9	38.8	50.4
D4	40.6	58.7	67.8	41.7	61.8	69.6	22.8	36.6	51.2
D2	28.4	42.3	51.3	23.2	35.4	44.5	40.3	56.4	68.2
D4+D2	44.0	62.7	72.1						
D3+D2				40.1	56.5	65.8			
D3+D4							33.7	42.2	58.4

and/or accessible over HTTPS. Our experiments show that the attack is robust to these factors. Next, we provide details on our findings with respect to each variable. For all experiments in this section, we used only automated traces.

Training and Testing Traces from Different Devices

Tables 3.6 and 3.7 summarize our result when using different smartphones for training and testing. Using one smartphone for training, and a different smartphone for testing led to a significant drop in identification accuracy. On the other hand, by training on two devices, and testing on a third, we were able to achieve identification accuracies above 80% with 6-second traces. This is likely because the classifier generalizes better when trained on multiple devices, which account for more variety within the traces.

Training and Testing using WiFi and LTE We collected power traces while accessing websites over both WiFi and LTE and experimented with three training-testing configurations: (1) LTE training and LTE testing, (2) LTE training and WiFi testing, and (3) WiFi training and LTE testing. Our results (see Table 3.5) show that accuracy

Table 3.8: Identification Accuracy (in %) using 70-day-old and 32-day-old training datasets collected using D1. All results were obtained using 125 features.

	Rank 1			Rank 5		
	2 s	4 s	6 s	2 s	4 s	6 s
32-day old, charged, w/ cache	8.0	11.9	13.0	24.7	27.8	32.3
70-day old, charged, w/ cache	3.6	3.7	2.2	13.8	20.2	24.3

obtained when training and testing on LTE is comparable to that of training and testing on WiFi (in Table 3.3). Further, though there was reduction in accuracies with WiFi training-LTE testing and LTE training-WiFi testing, the overall results were significantly higher than the baseline (2%) at Rank 1, and at least 73.9% for 6 second traces.

Aging of Training Traces Many of the webpages considered in this work contain content that changes over time. For example, the home page of `yahoo.com` shows recent news, and is therefore updated several times a day. For these webpages, a training dataset collected at a specific point in time might not be representative of the webpage's behavior at a later point, when the testing traces are collected. To determine the impact of aging on training data, we collected testing traces 32 and 70 days after training, with cache enabled and fully charged battery. The corresponding results are summarized in Table 3.8. For reference, all traces used to obtain the results reported in tables 3.3 and 3.4 were collected within a 48-hour timeframe.

When we trained the classifier on traces collected 32 days before testing, Rank 1 identification accuracy dropped below 13%. Even worse, the identification accuracy with training traces collected 70 days before testing was consistently below 4%. This suggests that, in order to achieve good identification accuracy, training traces should be updated frequently.

Foreign vs. Domestic Websites We tested this variable because the distance between the client and the host serving a webpage is known to affect packets' delay and jitter [63]. More specifically, the farther the host serving the content, the more variable will be its measured bandwidth and delay. In turn, this variability affects page loading, and hence the corresponding power traces.

Table 3.9: Rank 1 webpage identification accuracy (in %) for domestic (indicated as "Dom.") and foreign ("For.") websites. Traces were collected using D1. All results were obtained using 125 features.

	2 s		4 s		6 s	
	Dom.	For.	Dom.	For.	Dom.	For.
Charged, no cache	98.1	92.9	99.7	97.1	99.7	98.2
Charged, w/ cache	99.4	97.1	99.4	100	100	100
Charging (30%), no cache	82.5	83.9	92.5	89.3	96.4	93.9
Charging (30%), w/ cache	87.5	87.9	94.2	95.0	96.1	96.4

We determined whether a webpage was served from a host (e.g., a server, or a CDN) within the continental United States or outside the United States using the Whois databases from ARIN [64] and APNIC [65]. Our dataset is composed of 36 domestic websites, and 14 foreign websites.

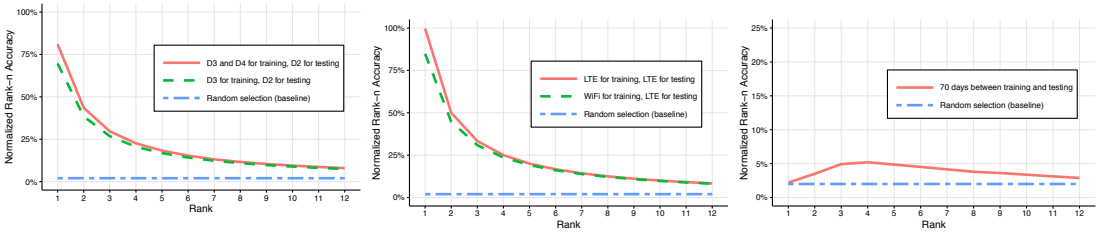
The results of our analysis are summarized in Table 3.9. Our experiments show that the location of the host serving a webpage has a very small impact on identification accuracy. Enabling cache led to a larger improvement in webpage identification accuracy for foreign websites than for domestic websites.

HTTPS vs. HTTP Websites We tested this variable because the use of encryption between the smartphone and the server can introduce noise in power traces. In particular, TLS requires additional communication rounds to exchange TLS session keys before a connection can be established. This can potentially increase the variability of power traces.

A total of 15 webpages in our dataset allow users to retrieve content using HTTPS, while the remaining 35 webpages only allow plain HTTP connections. Our results, reported in Table 3.10, show that there is no significant difference in identification accuracy between the two types of websites. This indicates that the attack is as effective for identifying securely transmitted webpages as with webpages transmitted without encryption.

Table 3.10: Rank 1 webpage identification accuracy (in %) for webpages retrieved via HTTPS and plain HTTP. Traces were collected using D1. All results were obtained using 125 features.

	2 s		4 s		6 s	
Scenario	HTTPS	HTTP	HTTPS	HTTP	HTTPS	HTTP
Charged, no cache	97.7	96.1	99.7	98.7	99.3	99.3
Charged, w/ cache	99.7	98.4	98.7	100	100	100
Charging (30%), no cache	81.7	83.4	90.0	92.3	95.0	96.0
Charging (30%), w/ cache	89.3	86.9	94.0	94.6	96.0	96.3



(a) Training on one or more devices, and testing on a different one.

(b) TTraining and testing performed using WiFi and LTE.

(c) Training and testing with traces collected 70 days apart.

Figure 3.6: Rank vs. Normalized Rank-n Accuracy plots show the results of our experiments with automated dataset, using frequency-domain features and classifier voting. All experiments were performed with a fully-charged battery and no cache. The plots in the top figure were obtained using 15 features, and the plots in the middle and bottom figures were obtained using 125 features, because these settings led to the best results.

3.6 Normalized Rank-n Accuracy

Figure 3.6 shows the Rank vs. Normalized Rank-n Accuracy plots for our classifier, obtained using frequency-domain features. All traces used in this evaluation were obtained with no cache, and with fully-charged battery.

Figures 3.6a and 3.6b show results of experiments where training and testing traces were from different devices, and different connectivity, respectively. The two figures show that increasing the rank leads to a decrease in normalized Rank-n accuracy. This means that the adversary has lower probability of correctly guessing a webpage as the rank increases. This is because Rank 1 accuracy is already substantially higher than the baseline. Therefore, increasing the rank provides almost no benefits towards correctly guessing the webpage. Although the normalized Rank-n accuracy approaches the baseline as the rank increases, figures 3.6a and 3.6b show that, even at Rank 12, our

technique still outperforms the baseline.

We evaluated the traces collected on a single smartphone, with training and testing data collected 70 days apart. This figure shows that when outdated traces are used for training, Rank 1 classification no longer provides the highest normalized Rank-n accuracy. This is because of two reasons: (1) Rank 1 accuracy in this setting is close to the baseline; and (2) as the rank increases, the classifier outputs a better uncertainty set compared to random choice. As a result, the Rank 2 to Rank 4 accuracies increase more than the uncertainty due to the increased rank. For instance, when going from Rank 1 to Rank 4, the output of the classifier includes three additional webpages that are far more likely to be correct than three pages chosen at random. Therefore, in this particular case, the adversary is more likely to correctly guess the webpage by first increasing the rank from 1 to 4.

3.7 Conclusion and Future work

In this work, we demonstrated that it is possible to accurately infer browsing activity on a smartphone using USB power consumption measurements. Our work is the first to study this side-channel attack on smartphones, and to analyze a multitude of factors that affect the traces that are collected during the attack, such as: battery charging level, user interaction with the touchscreen, trace length, time between collection of training and testing traces, WiFi and LTE connectivity, training and testing device mismatch, and website characteristics such as type of connection (HTTP or HTTPS) and location of the host serving the webpage relative to the smartphone.

We performed extensive experiments to validate our approach. Our results show that the attack successfully identifies webpages loaded using the standard Android mobile browser at least 91.7% of the times within four seconds for the automated dataset. For our user-actuated dataset, identification accuracy was between 54.2% and 88.4% with six seconds of power consumption data. Factors such as the availability of cache, the use of secure connections (HTTPS) and the location of the websites had a small

effect on identification accuracies. Other factors, such as mismatch between training and testing traces due to the use of different devices and the type of wireless connectivity negatively impacted accuracies. However, training on multiple devices allowed us to achieve accuracies substantially higher than the baseline for Rank 1.

Overall, our results show that the attack is highly effective, because webpage loading generates power signatures that are: (1) **distinctive**: different webpages generate different power traces due to factors such as the amount of data (text, images, and videos) being retrieved, the number of TCP connections required to retrieve all webpage components, and the computational cost of the scripts running within the webpage; and (2) **consistent**: each time a particular page is loaded, it generates a power trace that is similar to its previous power traces.

Though our work focuses on webpage identification, we believe that the same side-channel can also be used to detect other user activities, such as which application is currently being used, and the timing of touch events (including typing information). We consider this work as the first step towards a deeper understanding of the extent of the information leaked through power analysis of the USB port of a smartphone. Clearly, there are a multitude of factors that we did not address in this project, such as number of applications installed on the smartphone, background processes, network congestion, WiFi/LTE signal strength, and specific user interaction. We leave the analysis of these factors to future work.

Chapter 4

USB Side-channel Attack on Tor

4.1 Introduction

Tor [9] is an application-level low-latency anonymity network that enables anonymous communication between a client and arbitrary Internet destinations. Tor uses onion routers [10] hosted by volunteers to unlink the identity and geographical location of the client from the server, and to conceal the identity of the server to any adversary that can observe the client's network activity. Because of these properties, Tor is frequently used to provide anonymous connections to overcome communication restrictions and to evade censorship. Users rely on Tor to conceal their activities from governments, employers, and ISPs, since those might abuse, misuse, or accidentally leak sensitive information. Additionally, Tor can be used to improve privacy and security of Internet of Things (IoT) devices [66] [67]. For instance, if these devices access cloud services via Tor, the cloud provider is unable to determine which devices belong to the same household. Further, IoT devices can provide remote access by instantiating a Tor hidden service [68], thus concealing the identity and location of the owner of the devices.

Because Tor is accessed by millions of users [11], including smartphone users (Tor Orbot, for instance, has more than 10 million installs on Android phones [12]), the security of Tor is becoming increasingly important. Given prior work on the security of widely-available public USB charging stations [2], in this work we investigate whether a

malicious charging station can infer which websites are accessed by the Tor user while charging her smartphones. The ability to determine which website is being accessed through Tor using power consumption information, rather than through observation of network traffic, represents a hitherto unexplored and potentially devastating attack.

Accessing web content via Tor has significant effects on how each webpage is loaded. When users browse webpages using Tor, all requests and the corresponding responses are forwarded by three Tor relays in the Tor circuit. Each relay encrypts and decrypts all data in transit. Since the relays are geographically distributed, each packet must travel a long distance before reaching its destination, thus introducing large and variable network delays. Further, because Tor circuits are usually composed of entirely different sets of relays, the introduced delay adds further uncertainty and introduces inconsistency when loading the same webpage. The construction of Tor circuits consumes additional energy, thus adding background noise to the power traces for webpage loading.

To illustrate how Tor affects webpage loading, we measured the loading time of six webpages on a Samsung Galaxy S6 with and without Tor. We loaded each webpage 5 times. The results are shown in Figure 4.1. Using Tor not only greatly increased the loading time (by over 4 times on average); it also introduced a larger variation within the loading time. The average relative standard deviation of loading time was 40.54% with Tor and 21.98% without Tor. The effects of loading webpages with Tor are further reflected in the power traces. Figure 4.2 shows the power traces collected while loading the homepage of `google.com`. We compared the power trace when loading the same webpage directly and using Tor (on the same smartphone and mobile browser). When loading `google.com` without Tor, most energy consumption occurs within the first second. When using Tor, the energy consumption is spreaded across a longer period (the first 7 seconds). The appearance of such random power patterns leads us to ask whether it is possible to identify webpages based on power signatures when using Tor.

To answer this question, in this project we introduce a new attack on Tor. This attack enables a malicious charging station to identify which website is being visited via

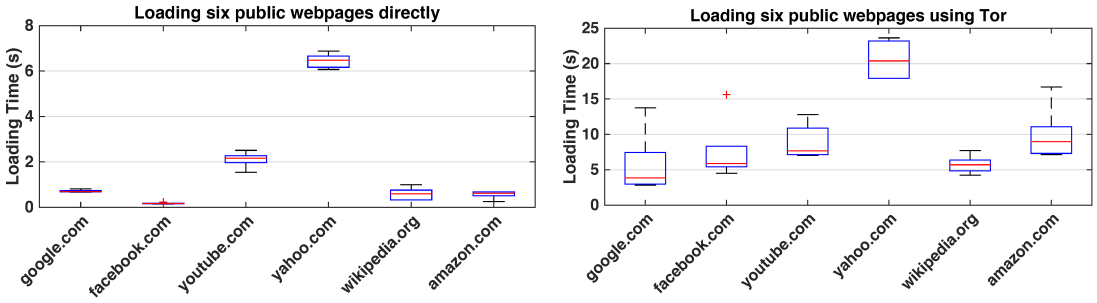


Figure 4.1: Loading time for six public webpages without Tor (left) vs. using Tor (right).

Tor by smartphone users. Our attack relies on power measurements performed while the user is charging her smartphone, and allows the adversary to determine which websites are visited. We were able to correctly identify websites accessed via the Orbot/Orfox Tor browser [69] with accuracies between 34.5% to 85.7% under realistic constraints, such as different network types (LTE and WiFi) and battery levels (30% to 50%, and 100%). Further, our attack was successful in identifying not only regular webpages, but also pages served by Tor hidden services, thereby increasing the scope of the threats identified in this work. We consider this a serious attack on Tor because (1) public charging stations are becoming widely available, making the attack scenario in this project very realistic and widespread, and (2) the level of privilege required to implement this attack is minimal, as it needs no access to (or manipulation of) network traffic, no malicious servers, and no exploitation of bugs in the Tor software. Because the security of Tor is critical to guarantee the safety and freedom of a large number of users around the world, any low-privilege attack that reliably and accurately infers user activity should be considered very seriously.

There are many studies on Tor attacks, which can be broadly categorized into passive attacks based on traffic analysis and active attacks against Tor network. For passive attacks based on traffic analysis, most studies are dedicated to either (1) website fingerprinting attacks [21-24], which enable an attacker to detect patterns indicative for webpages in Tor traffic, or (2) traffic confirmation attacks [25-31], which eavesdrop on both ends of a communication over a long period of time to deanonymize Tor users. As

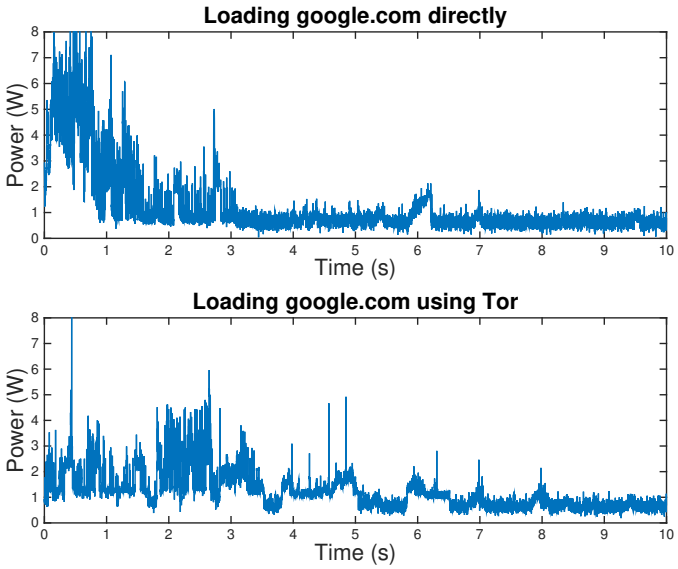


Figure 4.2: Power traces collected during the first 10 seconds of loading google.com without Tor (upper plot) vs. using Tor (lower plot). The x axis shows time from the beginning of the webpage loading, and the y axis shows the power drawn from the USB port.

for active attacks against Tor network, some studies are devoted to path-selection attacks based on network congestion [32]. Others exploit design flaws in Tor and propose DoS attacks to disable Tor nodes [33]. Compared to these attacks, our technique requires no privileged access to the network or exploitation of bugs in the Tor software. Particularly, our attack only depends on the power consumption data measured at the USB charging port.

Table 4.1: Fifty webpages selected from Alexa top non-adult websites (as of Dec 2016).

google.com	amazon.com	ebay.com	microsoft.com	fc2.com	sogou.com	wikia.com	dropbox.com	cnn.com
facebook.com	twitter.com	wordpress.com	vk.com	snapdeal.com	blogger.com	pixnet.net	whatsapp.com	espn.com
youtube.com	sina.cn	msn.com	apple.com	ask.com	naver.com	coccoc.com	nicovideo.jp	thepiratebay.org
yahoo.com	weibo.cn	pinterest.com	imdb.com	stackoverflow.com	mail.ru	adf.ly	rakuten.co.jp	daum.net
wikipedia.org	ok.ru	paypal.com	office.com	netflix.com	github.com	adobe.com	bbc.com	nytimes.com
dailymail.co.uk	stackexchange.com	booking.com	indeed.com	salesforce.com	———	———	———	———

Table 4.2: Fifty random selected hidden services (all with *.onion* as domain name suffix).

rougmnvswfsmd4dq	yuxv6qujajqvmypv	nql7pv7k32nnqor2	s5q54hfww56ov2xc	sblib3fk2gryb46d	qputrq3ejx42btla	4yjes6zfucnh7vcj	abbujjh5vqtq77wg
ityukvsoqjgzcimm	kxojy6ygju4h6lwn	cashis7ra6cy5vye	3g2upl4pq6kufc4m	fdwocbsnity6vzwd	zqktlwi4fecvo6ri	b34xhb2kjf3nbuyk	74ypjqjwf6oejmax
65px7xq64qrib2fx	fzqnrlcvhkgbdwx5	clockwise3rldkgu	libertygb2nyeyay	xmh57jrzrnw6insl	jmkxdr4djc3cpsei	djypjjvw532evfw3	76qugh5bey5gum7l
hss3uro2hsxfogfq	kpynyvym6xqi7wz2	fbcy5ylyoeqzqzcr	undergunbgzlc2ey	o6klk2vxlpunyqt6	ccxdnvotswsk2c3f	usjudr3c6ez6tesi	nare7pqnmnojs2pg
vu2wohoog2bytxgr	xfnwyig7olypdq5r	54ogum7gwxhtgiya	slwc4j5wkn3yyo5j	c3jemx2ube5v5zpg	flibustahezeous3	kbvbh4kdddiha2ht	tetatl6umgbmtv27
answerstedhctbek	tfwdi3izigxllure	gjobqjj7wyczbqie	ll6lardicrvrljvq	aaaajqiyzj34rhjm	hss3uro2hsxfogfq	w363zoq3ylux5rf5	grams7enufi7jmdl
drystagepmi5msdm	greendrgfjz7ks5f	———	———	———	———	———	———

4.2 Data collection

In this section, we first introduce the hardware, software, and network setup for the collection of power traces. We then explain how webpages and Tor circuits are selected. Finally, we present details on all datasets used in this project.

4.2.1 Experiment Setup

We powered the smartphone using a Rigol DP832 power supply [70]. It was set to 5.5 V when the smartphone battery was fully charged, and to 9 V when the smartphone battery was charging from 30%. The latter setting is supported by the Samsung Galaxy S6 and other smartphones compatible with Qualcomm Quick Charge [71], and resulted in a wider power consumption dynamic range. As suggested by USB charging specification [72], we modified the charging circuit so that the data pins (D+ and D−) of the USB cable were connected using a 200 Ω resistor to allow for charging currents above 500 mA. To measure the instantaneous smartphone power consumption from the USB port, we inserted a 0.1 Ω shunt resistor on the GND wire of the USB cable, and measured the voltage drop across the resistor using a National Instruments USB-6211 DAQ [73]. The DAQ was set to use a sampling rate of 200 kHz.

To collect power traces, we used Orbot and Orfox on two Samsung Galaxy S6 smartphones, denoted as phone A and phone B, respectively, in the rest of this project. Orbot implements a local proxy that provides access to the Tor network. Orfox is a web browser based on the smartphone version of Firefox. It enhances Firefox by including features that improve user privacy, such as HTTPS Everywhere [74]. Further, it disables the execution of JavaScript code by default. We connected Orfox to Tor using the Orbot instance on the smartphone. To collect data reliably, we modified Orfox by disabling the Android flag `FLAG_SECURE` to enable screenshot once a web page was loaded. This was used to manually verify that all pages were loaded successfully. We also implemented a Tor option that enables manual selection of the second relay, so as to create a "fixed" Tor circuit.

To load each webpage automatically, we developed an Android background service that cycled through our webpage list (see tables 4.1 and 4.2). After loading each webpage, the service paused for 12 seconds, and then logged the URL that was loaded, together with the corresponding timestamp.

We loaded all webpages using the WiFi network on the campus of The College of William & Mary, and via the T-Mobile LTE network in Williamsburg.

4.2.2 Datasets

In this subsection, we introduce Tor hidden service [68]. To collect data, we used two types of Tor circuits: fixed, and automatic. Details follow.

Collection of Data from Tor Hidden Services In contrast with servers on the public Internet, Tor hidden services are accessible only using the Tor network. Hidden service providers reside on Tor relays or Tor clients, and offer various services including web hosting, instant messaging, and SSH, while hiding the hidden service IP addresses. Each Tor hidden service hides behind several "introduction" relays in the Tor network. When visiting a hidden service, the Tor client first downloads the service's public descriptor (identified by a unique 16-character name followed by ".onion"). Then, it creates a Tor circuit to a randomly selected "rendezvous" relay, and sends the rendezvous relay's address to the hidden service through one introduction relay. The hidden service creates a Tor circuit to the rendezvous relay, and the client uses the "rendezvous" relay to exchange encrypted messages with the hidden service.

In this project, we refer to the webpages hosted on public Internet servers as "public webpage", and to web content hosted on hidden services as "hidden service".

We collected power traces while loading selected public webpages and hidden services. Tables 4.1 and 4.2 list all websites used in our experiments. For public webpages, we selected the home pages of the 50 most popular non-adult websites accessible via Tor, based on the Alexa ranking. We excluded public webpages that do not display content without JavaScript, because Orfox disables JavaScript by default. For hidden services,

we randomly selected 50 websites from The Hidden Wiki [75] that were consistently available during the experiments. Because 100 webpages represent only a small portion of the Web, we consider this work as a proof of concept. However, even with this restriction, our results conclusively show that substantial information is leaked when the adversary is able to monitor power consumption during page load.

Selection of Tor Circuits

When a Tor client builds a circuit, it first selects three relays from a public directory. The client then connects to the first relay ("entry"), and uses it to extend the circuit to the second relay ("middle"). The client finally uses the first two relays to extend the circuit to the last relay ("exit"). As it constructs the circuit, the client shares a unique symmetric key with each relay.

We performed our experiments using two types of Tor circuits: "automatic", and "fixed". For *automatic* circuits, we allowed Orbot to select a new circuit for each webpage loading using the default path selection protocol [76]. By default, the entry relay is selected among a small group of long-term entry servers (*guard* nodes), and it does not change for a relatively long time. However, Orbot settings allow the user to disable using entry guard by setting "UseEntryGuards" to 0. We used this option in our experiments to model the inability of the adversary to use the same Entry Guard as the user.

The circuit used to load a specific webpage changes every 10 minutes by default. Because in our data collection the time between collection of subsequent traces from the same webpage is larger than 30 minutes, power traces from the same website were collected using different circuits.

With the fixed circuits, we manually chose all three Tor nodes in the circuit, and used them to load all webpages. In our dataset, we denoted the circuit composed of `anonymiton` (in Germany, entry), `torfa` (in Hungary, middle), and `Hermes` (in France, exit) as "Cir-1". We denoted the following circuit as "Cir-2": `inky` (in Switzerland, entry), `cry` (in Netherlands, middle), and `hessel1` (in Romania, exit). Although in practice Orbot (or any modern Tor implementation) does not use a fixed circuit for

loading multiple webpages, we used this type of circuits to evaluate the scenario where training and testing data were collected under conditions that were as consistent as possible. This allowed us to quantify the loss of accuracy due to noise induced by the use of different Tor circuits in training and testing.

The datasets used in our experiments are listed in Table 4.3. Each dataset is composed of 40 power traces collected from the 100 webpages. Each trace has a duration of 10 seconds, because this allowed Orfox to load almost all webpages completely. We loaded all 100 webpages once, and then repeated this process 40 times. If a page did not load successfully, we replaced that trace with a new one at the end of the data collection session.

Table 4.3: Configurations used to collect power trace datasets

Dataset	Phone	Circuit	Network	Battery Level
1	A	Automatic	WiFi	100%
2	B	Automatic	WiFi	100%
3	A	Automatic	LTE	100%
4	B	Automatic	WiFi	30% to 50%
5	A	Fixed #1	WiFi	100%
6	B	Fixed #2	WiFi	100%
7[1]	A	Automatic	WiFi	100%
8	B	Automatic	WiFi	100%

4.3 Feature selection and classification

To identify the public webpages and hidden services loaded on the smartphone, we first extracted time- and frequency-domain features from the power traces, and then trained a Random Forest classifier on the resulting feature vectors. We used the trained classifier to predict the webpages on new power traces.

[1]To minimize the effect of time difference on the identification accuracy when comparing two different phones, we collected datasets #7 and #8 under the same scenario as datasets #1 and #2 but with shorter interval between collections of the two datasets.

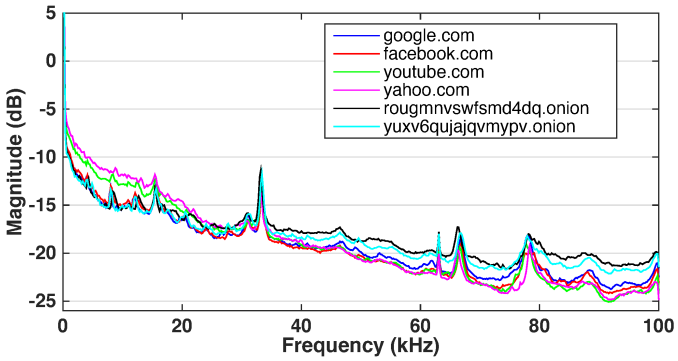


Figure 4.3: Spectrogram analysis on power traces sampled while loading six different websites.

4.3.1 Feature selection

We experimented using time-domain features, such as mean, RMS, and correlation coefficient, and frequency-domain features based on cepstrum analysis [77]. We found that features based on spectrogram analysis led to higher accuracies. Figure 4.3 illustrates that different webpages have distinctive frequency spectrum patterns, where the magnitude values in most frequencies are significantly different among these webpages. We further divided each power trace into several overlapping 0.5-second segments. We calculated the spectrogram of each segment (using window length of 1000 samples, and 50% overlap between windows). The spectrogram results contain the magnitudes of frequencies in the range of 0 Hz $\sim$ 100 kHz. We divided this range into 125 equal-size bins to reduce the effects of noise of individual frequencies, and calculated the average magnitude of all the frequencies in each bin as its corresponding feature, thus transforming each power trace segment into a feature vector of 125 elements.

4.3.2 Classification

The classification problem can be abstracted as follows. We use $X = (x_1, x_2, \ldots, x_p)$ to denote a feature vector with p features. Variable Y represents possible classes $1, 2, \ldots, K$. Given a training dataset S with N observations (X_i, Y_i), we first use S to train a classifier $\hat{C}(X) \in \{1, 2, \ldots, K\}$, and then use $\hat{C}(X)$ to predict the classes of testing feature vectors.

We used Random Forest for classifier training. A random forest consists of a set of decision trees. We use B to denote the number of decision trees to build for a random forest. There are three steps to train the random forest: (1) from dataset S, each time we randomly draw N observations with replacement to create a *bootstrap* dataset S_b for $b = 1, 2, \ldots, B$, (2) from each S_b, we train a decision tree $C(S_b, X)$, and (3) the random forest classifier is the ensemble of all $C(S_b, X)$, and it uses majority vote to make prediction on testing feature vectors.

In the following, we give details of the above step (2). Each decision tree is built from the root node. At each node, we randomly select a subset of all the p features, denoted by $F = \{x_1^*, x_2^*, \ldots, x_m^*\}, m = \lfloor log_2 p + 1 \rfloor$. For each feature $x_i^*, i = 1, 2, \ldots, m$, we use G_i to denote the set of all possible x_i^* values in the dataset. We try each possible test $x_i^* < g, g \in G_i$ to split the current node, and choose the test that generates the largest "information gain". To calculate the information gain, at each node, assuming P_i is the occurrence probability of class $i, i = 1, 2, \ldots, K$, we first calculate the *Shannon entropy* as:

$$H = -\sum_{i=1}^{K} -P_i \log_2 P_i \tag{4.1}$$

Assume that the entropy at current node is H. After a splitting, there are L percent of observations in the left child and R percent of observations in the right child. We use H_L and H_R to denote the entropy at the left and right child respectively. Then the average entropy after splitting is:

$$H_{After} = H_L \times L + H_R \times R \tag{4.2}$$

The *information gain* of a splitting is defined as $(H - H_{After})$. At each node, our goal is finding the splitting to:

$$maximize(H - H_{After}) \tag{4.3}$$

The above process recurses on each child until a stopping condition is satisfied, such as all observations in the node belong to the same class.

After all decision trees are trained, the random forest classifier is defined by:

$$\hat{C}(X) = majority\_vote\{C(S_b, X)\}, b = 1, 2, \ldots, B \tag{4.4}$$

We used the WEKA [78] implementation of Random Forests. For each of our experiment scenario, we used 20 power traces per webpage to train the classifier, and the other 20 power traces per webpage for testing. We trained the classifier using segments of all training traces. To identify a testing power trace, we first classified all the segments of this trace, and then used majority voting of these segments to determine the class of this trace.

4.4 Performance Evaluation

We first present the identification accuracies of our technique for the baseline scenario (i.e. using WiFi and fully-charged battery). We then discuss how different variables, including using different phones for training and testing, network types, and battery charging levels, affect identification accuracy.

4.4.1 Identification Accuracy for Baseline

We list the datasets corresponding to our baseline as #1 and #2 in Table 4.3. We evaluated the following three cases: (1) using all traces for both public webpages and hidden services; (2) using traces from public webpages for training and testing; (3) using traces from hidden services for training and testing. In each case, we used half traces for training, and the other half for testing. The results are shown in Table 4.4.

Table 4.4: Webpage identification accuracy using WiFi and 100%-charged battery (baseline)

Phone	All webpages	Public webpage only	Hidden service only
A	79.05%	76.3%	87.3%
B	85.7%	82.2%	89.2%

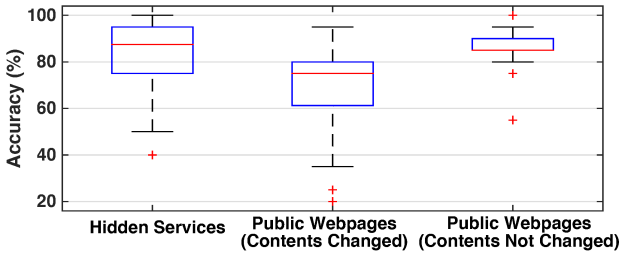


Figure 4.4: Identification accuracy comparison among (1) hidden services, (2) public webpages with content changes, and (3) public webpages without content changes.

We were able to identify hidden services with higher accuracy (87.3%) than public webpages (76.3%), as shown in Figure 4.4. There are two possible reasons for the accuracy difference between public webpages and hidden services. First, we measured the loading time of six hidden services (shown in Figure 4.5). We observed that, the loading time of hidden service is more consistent (19.82% relative standard deviation, on average, compared to 40.54% for public webpages) and has smaller range (from 3.25 s to 10.63 s, compared to 2.81 s to 23.63 s for public webpages). Possible reasons for these differences include: (1) hidden services are mostly simple static webpages; and (2) their contents rarely change over a long time. In contrast, public webpages usually contain large-size elements. Since public webpages need longer loading time than hidden services, they have a higher chance of being influenced by the instability of Tor circuits. As a result, there are more inconsistency and noise in the training and testing power traces of public webpages.

Second, we examined the screenshots for each webpage loading, and found none of the 50 hidden services changed their displayed contents during the data collection period (about 1 day). In comparison, 31 of the 50 public webpages displayed different contents during the collection, which further affects the consistency among power traces of public webpages. Figure 4.4 shows that public webpages without content changes have higher average accuracy than public webpages with content changes.

The content changes of public webpages are either due to frequent website updates (e.g. for news websites), or because different versions of the webpage were loaded based

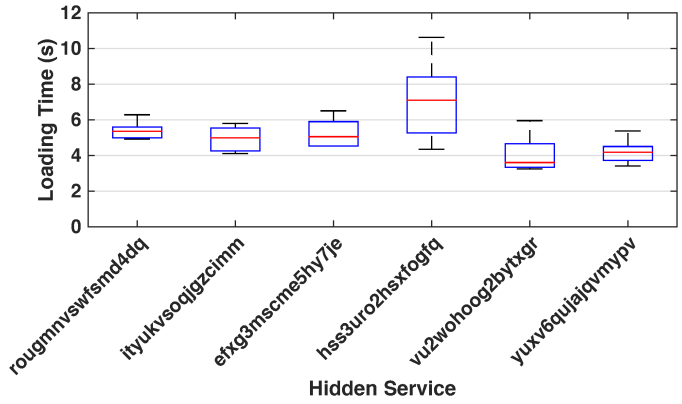


Figure 4.5: Loading time for six hidden services using Tor.

on the geographical location of the Tor exit relay. For example, we checked the four public webpages with identification accuracy lower than 40%, and found two of them were loaded with different versions (based on website language, content, and layout): there were 9 versions for `paypal.com` (with accuracy of 30%), and three versions for `dropbox.com`. (with accuracy of 25%).

To improve the identification accuracy, we tried to use a specific version of these websites for training and testing. For example, we used the traces for `paypal.com/de` instead of `paypal.com` and re-conducted the whole training and testing. The identification accuracy was improved from 30% to 90% for this specific webpage, and increased from 79.05% to 79.65% for all webpages.

4.4.2 Impact of Phones

When we used the dataset collected from phone A to train the model, and tested the dataset from phone B, the original classification model didn't work well and we improved it in the following way. By examining the spectrograms of power traces from different phones, we observed that beyond a specific frequency (about 3 KHz), the difference in magnitude distribution among spectrograms is more determined by the phone than by the webpage. Thus we increased the frequency resolution of the spectrogram, and used the magnitudes of the first 250 frequency points (ranging from 0 Hz to 3039.6 Hz) as the feature vector for each power trace. We then used Sequential Minimal Optimization

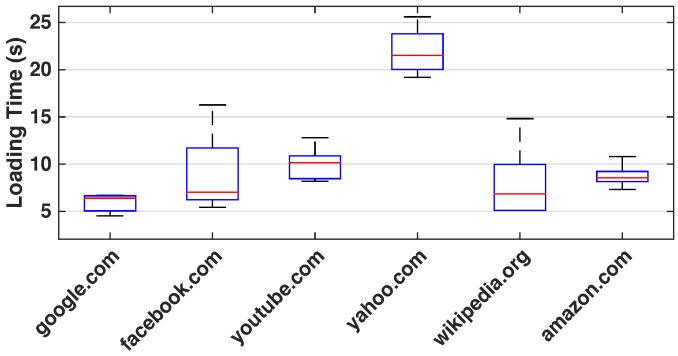


Figure 4.6: Loading time of public webpages using LTE network.

(SMO) algorithm [79] for model training and testing. The identification accuracies using two phones are presented in Table 4.5.

Table 4.5: Webpage identification accuracy using different phones for training and testing

Train	Test	All webpages	Public webpage only	Hidden service only
A	B	43.13%	48.75%	47.05%
B	A	36.58%	43.2%	40.35%

Even though the accuracy decreases when training and testing on different smartphones (36.58% to 43.13% for all webpages using two phones, compared to 79.05% to 85.7% using the same smartphone for training and testing), it is still significantly higher than that of random chance at 1%.

4.4.3 Impact of Network Characteristics

In dataset #3, we collected training and testing traces using LTE network. The results in Table 4.6 show that the identification accuracy when training and testing on LTE (71.75%) is worse than that when using WiFi (79.05%, see Table 4.4). Consistently with our experiments based on WiFi, we observed that the accuracy for public webpages is lower than that of hidden services, and the accuracy decrease for public webpages (64.8% using LTE, compared to 76.3% using WiFi) is larger than the decrease for hidden services (78.7% using LTE, compared to 87.3% using WiFi).

Table 4.6: Webpage identification accuracy using LTE

All webpages	Public webpage only	Hidden service only
71.75%	64.8%	78.7%

One possible explanation is that LTE network contributes to additional noise in the power traces, compared to WiFi. We measured the loading time of six public webpages using LTE. The results are shown in Figure 4.6. Compared with the results of using WiFi , the average loading time increased by 9.1%, which indicates that LTE introduced more unpredictable delays than WiFi.

We also trained the model with WiFi traces, and tested it using LTE traces, and vice versa. The identification accuracies shown in Table 4.7 are significantly lower than that when using LTE traces for both training and testing. This indicates that the adversary needs to train a different model for each network.

Table 4.7: Cross testing using LTE and WiFi networks

Train	Test	All webpages	Public webpage only	Hidden service only
WiFi	LTE	24.55%	24.25%	42.9%
LTE	WiFi	21.8%	21%	28.15%

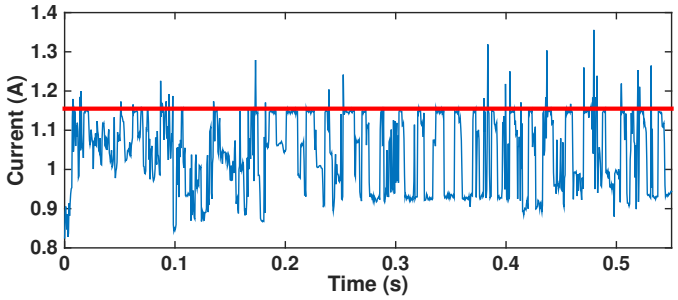


Figure 4.7: Capped current when smartphone battery is partially charged.

4.4.4 Impact of Battery Charging level

When the smartphone is charging, a large part of power is used to charge the battery. In contrast, when the battery is fully charged, almost all current from the charger is

used to power the phone, including loading the webpage. Thus, battery charging level impacts the amount of information that can be inferred from the power trace on webpage loading. We collected the power traces for 30% to 50% battery level. Before each round of the collection, we discharged the phone battery to 30%. Then we shuffled the 100 webpages into a random sequence, and collected one trace for each webpage following this sequence. After collecting all 100 traces in one round, the battery level increased to about 50%. Then we discharged the battery to 30% again, and repeated the above collection process. This process was repeated 40 times in total. Dataset #4 includes the traces collected while the smartphone charging level was between 30% and 50%.

We used half of dataset #4 for training, and the other half for testing. The identification results are in Table 4.8. We observed that the accuracy decreases sharply when the battery is not fully charged. One reason is that the maximum charging current is capped by an upper limit (1.15 A in this case), which is imposed by the smartphone's charging circuit (see Figure 4.7), where 98.17% of the samples in the power trace have current value below 1.15 A. This limitation distorts the power signals and decreases the effectiveness of our technique. Further, we found that there were strong noisy signals periodically appearing in each trace. The same signals did not appear in the power traces collected when the battery was fully charged.

Table 4.8: Webpage identification accuracy when battery level is from 30% to 50%

All webpages	Public webpage only	Hidden service only
36%	34.5%	46%

4.4.5 Impact of Tor Circuits Type

We evaluated the scenario in which training and testing data were collected using the same *fixed* circuit. The corresponding datasets are indicated as #5 and #6 in Table 4.3. Table 4.9 reports the identification accuracies when using fixed circuits "Cir-2" and "Cir-2" on different phones. Compared to the results in Table 4.4 obtained using automatic circuits, accuracies are not significantly higher, or are even lower in some cases. One

possible explanation is that the throughput of each relay in the fixed circuits are always changing. This adds unpredictable delays for each webpage loading and increases the inconsistency between power traces for training and testing. In our experiments, we observed that during some periods the fixed circuits could not be used to load any webpages at all. In contrast, automatic circuits are constructed using optimized path selection protocol. In practice, our experiments show that the adversary should use automatic circuits to collect training traces.

Table 4.9: Webpage identification accuracy using fixed Tor circuits

Phone	Circuit	All webpages	Public webpage only	Hidden service only
A	#1	78.1%	79.6%	84.8%
B	#2	82.6%	85.7%	86.1%

4.5 Conclusion and Future Work

In this work, we demonstrated a technique based on USB power analysis that allows a malicious charging station to identify which webpages are loaded on a smartphone using Tor. To our knowledge, this is the first work to study attacks on Tor based on smartphone power side-channels.

We validated our attack under realistic smartphone constraints by collecting and analyzing power traces under several scenarios, including different networks (WiFi and LTE), different devices, and different battery charging levels.

We correctly identified webpages visited using the official mobile Tor browser. We achieved accuracies between 36.58% and 85.7% when the battery was fully charged, and between 34.5% and 46% when the battery level was at 30%-50%. In comparison, accuracy obtained by random website selection is 1% for all websites, and 2% when considering hidden services or public webpages alone.

We consider this work the first step towards a full characterization of power side-channel attacks on Tor. As such, there are several variables that were not taken into

consideration in this work. For instance, future work will include a large webpage dataset to fully characterize how our technique extend to the entire web. Further, we will investigate how to to improve the identification accuracy for the scenario when battery charging level is low.

Chapter 5

MEG: Memory and Energy Efficient Garbled Circuit Evaluation On Smartphones

5.1 Introduction

Secure computation is the process of multiple parties computing the value of a function without involving any outside parties. To protect the privacy of computation participants, the input of each party can not be leaked to any other parties. Secure computation has wide usage in areas such as electronic voting [80], privacy-preserving data mining [81], and biometric-based authentication [49], [82]. As various Internet-of-things (IoT) devices are increasingly being deployed, their users' privacy can also be protected by secure computation. For example, when an IoT device is part of a crowdsourcing system and it needs to exchange private data with other parties, secure computation can be used to prevent the IoT device from losing data privacy to malicious parties, while still allowing it to exchange data with other crowdsourcing participants and obtain the final computation results [83], [84].

When there are two parties involved in secure computation, garbled circuit protocol [85] is a popular solution. The idea of garbled circuit is that one party ("generator")

transforms a function into a boolean circuit composed of wired binary gates, and then "garbles" this circuit by assigning encryption keys to represent possible values (0 or 1) at each wire. The circuit generator sends the garbled circuit, as well as its garbled input, to the other party ("evaluator"). The evaluator computes the output keys and then decodes the function output. When considering practical functions to evaluate, the generated garbled circuits usually contain a large amount of circuit gates. For example, in this work we need more than 30,000 gates to implement AES and more than 9 million gates to compute edit distance using garbled circuit. Therefore, the circuit evaluation process requires substantial data transmission and computation workload, which causes significant memory and energy consumption on the circuit evaluation device. On resource-limited mobile devices, such as smartphones and IoT devices with small RAM capacity, low CPU performance, and short battery life, garbled circuit evaluation has only become practical in recent years [45]. To increase the scalability of garbled circuit evaluation on mobile devices, it is crucial to improve the memory and energy efficiency.

When considering the method of data transmission between the circuit generator and the evaluator, existing garbled circuit solutions can be grouped into two types: (1) non-pipelining, and (2) pipelining. In the non-pipelining method, the evaluator stores the entire garbled circuit in its memory before evaluation. Because the transmission size of garbled circuits can be very huge for practical functions, non-pipelining method brings high memory overhead because all the transmitted data need to be stored in the RAM. In the pipelining method ([90], [91], [92], [42]), the generator sends each garbled gate as soon as possible. The evaluator only needs to keep one received gate, and deletes it from the memory as soon as it's evaluated. As a result, the evaluator only needs a small size of memory to store the current gate for evaluation. However, the pipelining method consumes more energy than the non-pipelining method (as shown in Section 5.4), because there are continuous data transmission and computation across the whole circuit evaluation process, which prevents the processor and network interface (such as WiFi module) to enter power saving state. On the contrary, non-pipelining method allows

the CPU to enter low power state during the data receiving process, and it enables the network interface to enter power saving mode during the circuit computation.

Contributions. In this work, we propose *M*emory and *E*nergy efficient *G*arbled (MEG) circuit evaluation method , which manages to combine the advantages of both non-pipelining (low energy consumption) and pipelining (low memory overhead) methods. The basic idea of MEG is to finish a middle ground between current techniques that offers the benefits of both. Specifically, MEG utilizes batch transmission of the circuit gates and multi-threading execution of gates transmission and calculation. In this way, MEG saves memory usage by only storing the current batch of gates. MEG also reduces energy consumption because it enables WiFi to enter power saving mode between two consecutive bursts of data transmission. To our best knowledge, we are the first to apply batching technique in garbled circuit evaluation for memory and energy efficiency.

We implemented MEG on smartphone (as circuit evaluator) as well as on PC (as circuit generator). We evaluated MEG using various burst sizes and compared its memory and energy consumption with existing circuit evaluation methods. We also compared the single-thread and multi-thread implementations of MEG. For memory consumption, MEG uses a fixed amount of memory regardless of the size of the circuit, which is just slightly above the pipelined version. For energy consumption, our evaluation results show that MEG consumes up to 42.1% less energy than the pipelining method and is close to the non-pipelining method. MEG also significantly reduces up to 56.7% running time compared to the pipelining method.

Security model. We assume both parties (generator and evaluator) in MEG are possibly *malicious* adversaries, who would arbitrarily violate the protocol to get other parties' information. Malicious generator could generates garbled circuit that can be used to get the evaluator's input. The security of MEG is the same as that of the underlying garbled circuit protocol. Because MEG is built on the cut-and-choose method [44], which garbles multiple copies of the same function to defend malicious circuit generation, it is secure against malicious adversaries.

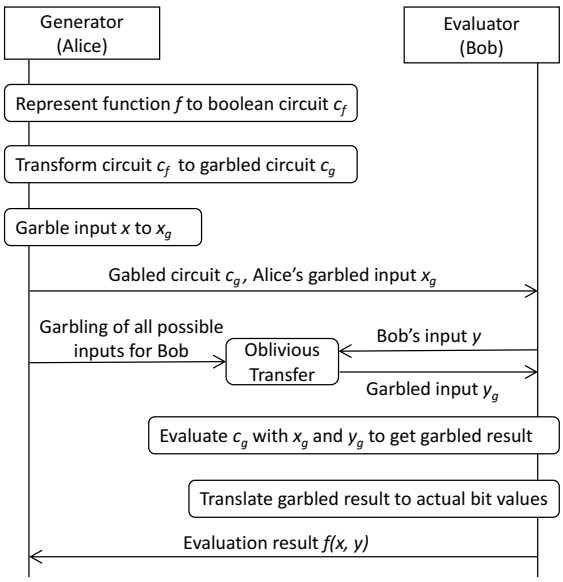


Figure 5.1: Basic garbled circuit evaluation process

The rest of this chapter is structured as follows. In Section 5.2, we give details about garbled circuit protocol and the problem of existing implementations. In Section 5.3, we propose the system design of MEG. In Section 5.4, we present the evaluation of MEG on smartphones. We conclude in Section 5.5.

5.2 Background and Motivation

In this section, We first give details about garbled circuit evaluation, then present two different evaluation methods: non-pipelining and pipelining. We explain the problems of current methods concerning memory and energy consumption.

5.2.1 Basic Procedure of Garbled Circuit Protocol

In garbled circuit protocol, there are two communication parties: circuit generator (*Alice*) and evaluator (*Bob*). Given a function f, *Alice* has input x_1, and *Bob* has input x_2. They want to compute the function result $f(x_1, x_2)$ only by themselves, without letting *Alice* know x_2 or *Bob* know x_1. The basic garbled circuit protocol uses the following six steps to calculate $f(x_1, x_2)$, as shown in Figure 5.1.

1. *Alice* represents the function f into a boolean circuit c_f, which only consists of binary gates (e.g. AND, OR, and XOR). Each gate is represented by a truth table consisting of four entries. Every entry describes the possible boolean values for the gate's two input wires (*input columns*) and one output wire (*output column*).

2. *Alice* turns circuit c_f into a "garbled" version c_g in the following way. Gate by gate, *Alice* generates an encryption key for each possible value of each input or output wire, and then encrypts the output column of each truth table entry using the corresponding input wire keys. *Alice* also re-orders all the entries in the garbled truth table.

3. *Alice* garbles her boolean value input x to x_g by converting each bit of x into the encryption key of the corresponding input wire defined in the garbled circuit c_g.

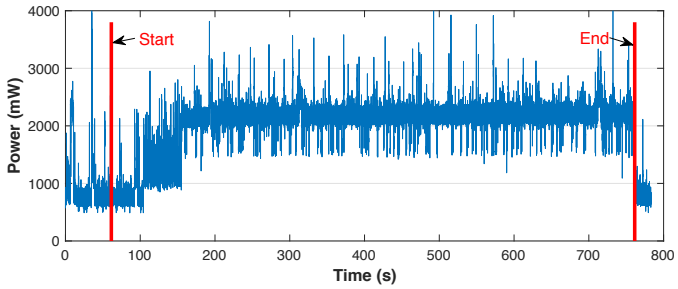
4. *Alice* sends the garbled circuit c_g and its garbled input x_g to *Bob*.

5. *Alice* and *Bob* run oblivious transfer protocol [88], so that *Bob* receives the garbled version y_g of his own input y, without disclosing y to *Alice*.

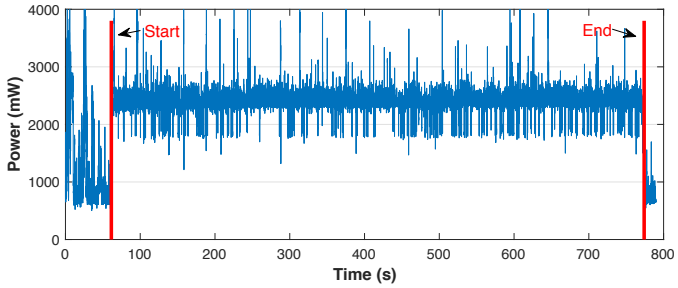
6. *Bob* evaluates each gate by using the two input keys he learned to compute the output keys. *Bob* decodes the final output keys into the corresponding bit value, and sends back the evaluation result to *Alice*.

There are several existing performance optimization methods for the above protocol. When constructing the garbled circuit, free-XOR technique [40] allows XOR gates to be evaluated without using the corresponding garbled tables or the cryptographic operations, thus eliminating some overhead for data transmission and processing. Furthermore, garbled row deduction [41] helps to reduce the size of each gate truth table by one entry, thus the communication and computation cost of circuit evaluation is further reduced.

To be secure against malicious adversaries, cut-and-choose method [44] can be used to enhance the basic garbled circuit protocol. The ideas of cut-and choose is as follows. First, the generator constructs multiple garbled versions of the circuit, and the evaluator randomly selects a set of the circuits (named "check circuits"), and challenges the generator to prove these circuits are validly garbled. If no malicious garbling is found within



(a) Non-pipelining



(b) Pipelining

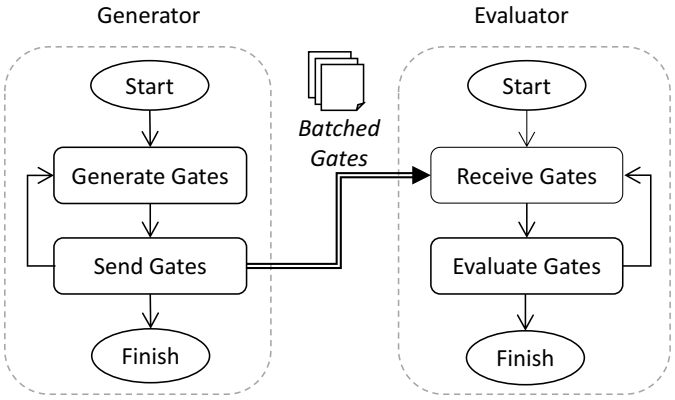
Figure 5.2: Power traces of non-pipelining vs. pipelining garbled circuit evaluation process on smartphone. The traces between "Start" and "End" are for circuit evaluation. The circuit is to compute 256-bit edit distance.

the check-circuits, the evaluator will evaluate the remaining unchecked circuits (named "evaluation circuits"), and use the majority of the evaluation results as the final output.

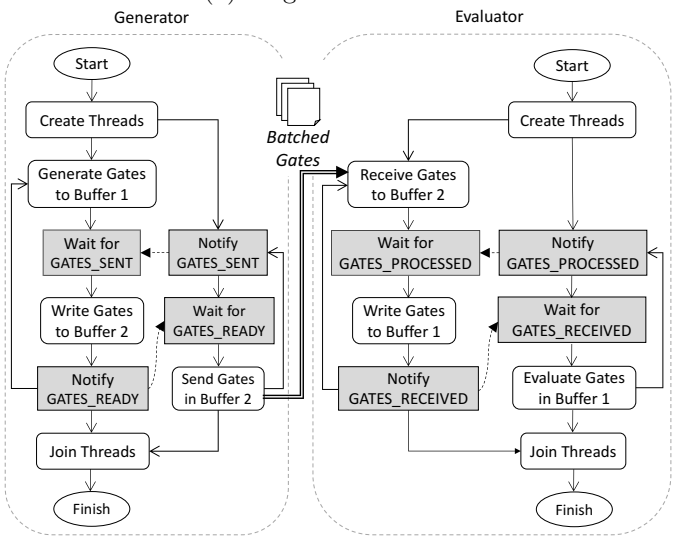
5.2.2 Pipelining Method

In "non-pipelined" implementations of garbled circuit (such as [89]), the circuit evaluation does not begin until the whole circuit is loaded into the evaluator's memory. This implementation method brings significant memory overhead to the evaluation device, especially for mobile devices with limited memory capacity. It also severely impacts the scalability of garbled circuit evaluation, because many practical functions requires garbled circuits of large size. The problem with large size is that the circuit has to fit in memory.

Pipelining evaluation method was proposed to solve the memory overhead problem. The premise of pipelining method is that there is no need to store the entire circuit in the RAM of the evaluator. Instead, the generation and evaluation of garbled gates can be



(a) Single-thread MEG



(b) Multi-thread MEG

Figure 5.3: Workflows of single-thread and multi-thread versions of MEG.

pipelined (i.e. overlapped in time). When the evaluator receives one gabled gate, it finds the corresponding gate of the circuit and begins to evaluate it as soon as the necessary inputs become ready. Gates already evaluated will be deleted from the memory. As a result, pipelining method significantly decreases the memory consumption from as large as hundreds of megabytes to only several bytes (determined by the size of one gate).

5.2.3 Energy Cost of Pipelining

Despite of the advantage of memory efficiency , pipelining evaluation consumes more energy than the non-pipelining method. This problem arises from the continuous data

reception and computation activities across the whole pipelined circuit evaluation process, which keeps both the CPU and the network interface in high power state most of the time. On the contrary, in non-pipelined circuit evaluation each phase (OT, sending circuit, evaluating the circuit) is performed at different times, and therefore the resources used at a particular time are used 100%, and are off during the other times. As a result, non-pipelining method enables the CPU enter low power state during the data transmission intensive period, and the network device enter power saving mode during the computation intensive process. For example, Figure 5.2 presents the power traces when using non-pipelining and pipelining method respectively to evaluate the garbled circuit for 256-bit edit distance calculation. We found that the average power consumption when using pipelining method (2376.89 mW minus baseline) is 30.6% larger than using non-pipelining method (1988.75 mW minus baseline). We also observed that pipelined method takes more time than non-pipelined. One reason is when the cpu is done evaluating one gate, it needs to request and wait for the next one, thus increases the total time due to process switching and network delay. As a result, the energy consumption using the pipelining method is 32% higher than using the non-pipelining method.

5.3 Design of MEG

Three techniques are used in the design of MEG: gates batching, slow start, and multithreading.

5.3.1 Gates batching and burst transmission

The idea of gates batching is that the circuit generator does not immediately send each generated garbled gate, but stores it to a buffer in the memory. Once the buffer size reaches a pre-set value (*burst size*), the generator sends the *batch* of gates in the buffer as a transmission *burst* to the evaluator. On the evaluator side, it also keeps a memory buffer to store the incoming batch of gates. After a batch of gates are received, the evaluator evaluates these gates and deletes them from the memory once the evaluation

is completed. Gates batching brings two advantages. First, because the buffer size is fixed and it does not depend on the size of the circuit, the memory consumption for circuit evaluation could be significantly reduced. Second, burst transmission saves energy consumption, because it allows the CPU to enter low power mode during the transmission of each batch, and the network interface enter power saving mode during circuit computation.

5.3.2 Multi-threading

In existing implementations of garbled circuit on smartphones (such as [42]), the generator only uses a single thread to process both generation and sending of the gates. On the other side, the evaluator also only creates one thread to handle both receiving and calculation of the gates. Figure 5.3a presents the single-thread version of MEG. The problem of using single-thread is that even some operations (such as date transmission and calculation) could run in parallel, they are forced to run in sequence within the same thread. As a result, for a circuit generator using the single-thread method, gates transmission needs to wait for gates generation to finish, and then gates generation will be paused until all buffered gates are sent to the evaluator. The single-thread circuit evaluator has the same problem, which introduces unnecessary process waiting time and increases the total execution time of circuit evaluation.

Because most smartphones use multi-core CPUs [93] and enable multi-thread processing, we designed a multi-thread version of MEG as shown in Figure 5.3b. Both the circuit generator and the evaluator create two threads. On the generator, one thread is used for gates generation, and the other thread is for batched gates transmission. These two threads use two signals (GATES_READY and GATES_SENT) to notify each other about current running status and synchronize. On the evaluator, one thread is used for gates receiving, and the other is for gates evaluation. They also use two signals (GATES_RECEIVED and GATES_PROCESSED) to synchronize. After evaluating the current batch of gates, the evaluator doesn't need to pause and wait for receiving a new

batch from the generator, thus decreasing the total time of circuit evaluation.

5.3.3 Slow Start

We uses slow start to further reduce the circuit evaluation time when using multi-thread MEG. When receiving batched gates, one problem is that when the batch size is large (e.g. 16 MB), the evaluator will experience long waiting duration for the first batch of data. For example, the waiting time for the first batch of 16 MB data is more than 5 s when using the average mobile internet download speed in US. Inspired by the slow-start technique in TCP protocol, we used a similar method to control the sizes of the first several batches. We set the initial batch size to be 0.25 MB and doubled it for each following batch, until it reaches the pre-set size. For example, if the batch size is 16 MB, then the sizes of the begining batches are {0.25, 0.5, 1, 2, 4, 8, 16, 16...} MB in sequence. In this way, the evaluator only needs to wait for about 0.1 s (still using 22.7 Mbps as the transmission speed) for the first batch of gates.

5.4 Evaluation

In this section, we evaluate the energy consumption and performance of MEG using a smartphone as evaluator. We also compared MEG with existing approaches.

5.4.1 Methods to compare

We compared the performance of the following garbled circuit evaluation methods:

(1) *Non-pipelining*: the circuit generator transmits the whole garbled circuit to the evaluator before evaluation begins.

(2) *Pipelining*. The evaluator only transmits one gate each time to the evaluator, and the evaluator beings to evaluate this gate as soon as inputs are available.

(3) *Pipelining+*. This is our improved pipelining implementation, which removes the useless 4-byte field containing the packet length in each data packet for one gate.

(4) *Single-thread MEG*. This is the single-thread implementation of MEG.

Table 5.1: Information of evaluated circuits

	Circuit function	
	AES-128	EDT-256
Total number of binary gates	34,136	9,959,904
Total circuit size	343.83 KB	80.27 MB
Number of evaluation-circuits for cut-and-choose	60	1
Total transmission Size	20.63 MB	80.27 MB

(5) *Multi-thread MEG*. This is the multi-thread version of MEG.

For both single-thread MEG and multi-thread MEG, we considered six different burst sizes in bytes: 0.5M, 1M, 2M, 4M, 8M, and 16M.

5.4.2 Experiment setting

The smartphone we used as circuit evaluator is a Samsung Galaxy S4 running Android 5.0. It has a Qualcomm Snapdragon 1.9GHz 4-core processor and 2GB RAM. For the circuit generator, we used a Thinkpad T440p PC with Intel i7 quad-core 2.4GHz CPU and 12GB RAM, and it runs Ubuntu 16.04. We created a controlled 802.11n WiFi network using a Netgear WNDR3700v2 router as the access point. Both the the smartphone (evaluator) and the PC (generator) connect to this access point through WiFi.

We used a Monsoon Power Monitor [94] to measure the power trace of the smartphone. The smartphone is powered totally by the Power Monitor instead of its own battery. The Power Monitor logs the power data it outputs to the smartphone in 5 KHz frequency. We analyzed the power trace offline to get both the time stamps (with resolution of 1 ms) and the power consumption at each time point (in mW), from which we calculated the total energy consumption for each power trace.

5.4.3 Circuits

We evaluated MEG on two circuits: (1) AES-128 for AES encryption, where the generator's input is a 128-bit key (E_k) and the evaluator's input is a 128-bit block to encrypt using E_k, and (2) EDT-256 for edit distance calculation, where the input of both the generator and the evaluator is a 256-bit binary string, respectively. The detailed information

of each circuit is shown in Table 5.1.

When using cut-and-choose, the total number of evaluation-circuits copies is 60 for AES-128 and 1 for EDT-256. As a result, the total transmitted circuit data size is 20.63 MB for AES-128, and 80.27 MB for EDT-256.

We implemented the five garbled circuits methods based on the code of Kreuter et al. [43] , which already implement pipelining circuit evaluation (including codes for oblivious transfer and cut-and-choose). We added implementations of gates batching, slow start, and both single-thread and multi-thread MEG. The circuit generator implementation on PC is written in C++, and the evaluator implementation on the smartphone uses Android NDK to encapsulate the C++ code.

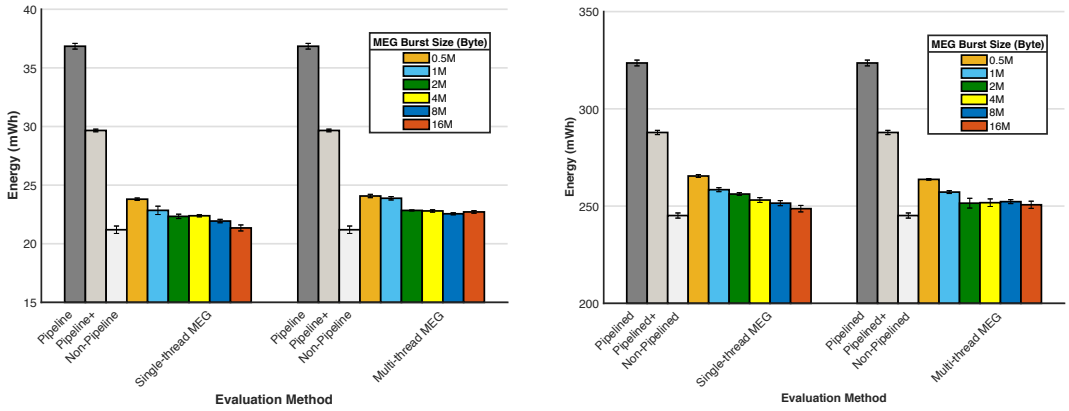
For both AES-128 and EDT-256, we reported the energy consumption and elapsed time on the smartphone using different evaluation methods. To get the *net* energy consumption for circuit evaluation, we measured the average power consumption of the smartphone while idle to calculate the background energy consumption, and then subtracted the background energy value from the total energy consumption. Due to the low variance between experiments, we carried out 5 measurements for each circuit type and evaluation method combination (for single-thread or multi-thread MEG, we also measured with different burst sizes).

5.4.4 Results

In the following sections, we compare the energy and memory consumption, as well as execution time of MEG with other methods.

5.4.4.1 Energy consumption

The energy consumption of all five methods are shown in Figure 5.4. The energy saving or overhead of MEG compared with other methods are listed in Table 5.2. We observe that compared to Pipelined method, MEG reduces the energy consumption by 35% to 42% for AES-128, and by 18% to 23% for EDT-256, respectively. Compared to Pipelined+,



(a) AES-128 (b) EDT-256

Figure 5.4: Energy consumption of MEG with different burst sizes, in comparison with other methods.

Table 5.2: Energy consumption saving of MEG compared with Pipelined and Pipelined+, and overhead compared with Non-Pipelined

(a) AES-128

	Single-thread MEG Burst Size (MB)						Multi-thread MEG Burst Size (MB)					
	0.5	1	2	4	8	16	0.5	1	2	4	8	16
Pipelined	-35%	-38%	-39%	-39%	-40%	-42%	-35%	-35%	-38%	-38%	-39%	-38%
Pipelined+	-20%	-23%	-25%	-25%	-26%	-28%	-19%	-19%	-23%	-23%	-24%	-23%
Non-Pipelined	12%	8%	5%	6%	4%	1%	14%	13%	8%	8%	6%	7%

(b) EDT-256

	Single-thread MEG Burst Size (MB)						Multi-thread MEG Burst Size (MB)					
	0.5	1	2	4	8	16	0.5	1	2	4	8	16
Pipelined	-18%	-20%	-21%	-22%	-22%	-23%	-18%	-21%	-22%	-22%	-22%	-23%
Pipelined+	-8%	-10%	-11%	-12%	-13%	-14%	-8%	-11%	-13%	-13%	-12%	-13%
Non-Pipelined	8%	5%	5%	3%	3%	1%	8%	5%	3%	3%	3%	2%

MEG decreases the energy consumption by 19% to 28% for AES-128, and by 8% to 14% for EDT-256, respectively.

When the burst size increases to above 2 MB, the energy consumption of MEG is very close to that of Non-Pipelined method, with only 1% to 8% overhead for AES-128, and 1% to 5% overhead for EDT-256.

We also observe that single-thread MEG and multi-thread MEG have close energy consumption. When the burst size increases, the energy consumption of both versions of MEG has a slight decreasing trend.

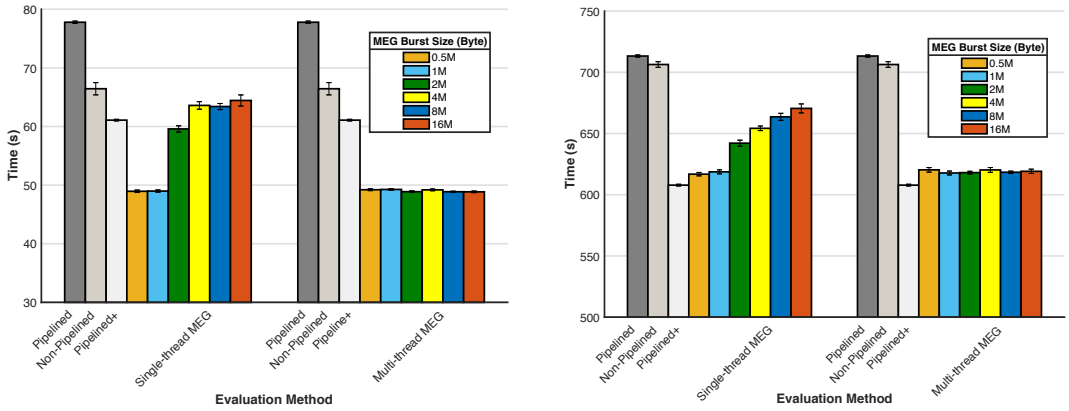
5.4.4.2 Memory consumption

The memory consumption of MEG is determined by the burst size, which ranges from 0.5 MB to 16 MB in our evaluation. Compared to the non-pipelined method that needs to store the whole circuit of large size (e.g. EDT-256) in memory, MEG significantly decreases the memory requirement. For example, based on the energy consumption evaluation, MEG can achieve satisfactory energy efficiency using 2 MB as the burst size. Compared with the non-pipelined method, MEG decreases the memory consumption by 97.5% (from 80.27 MB to 2 MB) for EDT-256. This shows that it is possible to obtain a substantial decrease in memory requirements compared to the non-pipelined method, while incurring in a negligible increase in energy consumption.

5.4.4.3 Execution Time

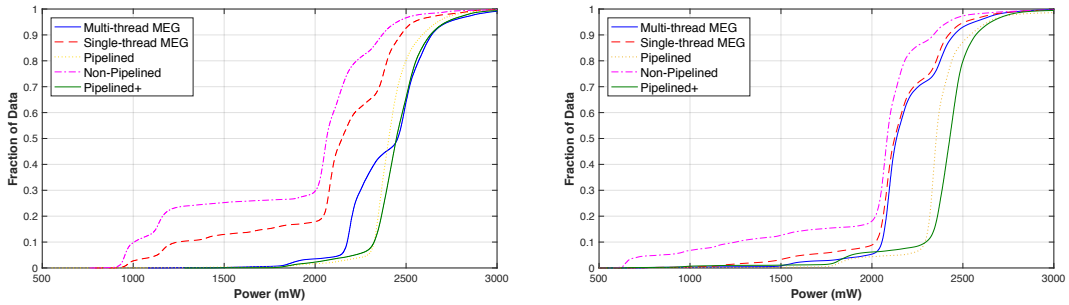
The execution time for circuit evaluation of all five methods are shown in Figure 5.5. We observe that the execution time for multi-thread MEG is consistent across all the different burst sizes. For AES-128, multi-thread MEG needs less time than all other methods: 56.7% decrease to Pipelined, 25.8% decrease to Non-Pipelined, and 19.8% decrease to Pipelined+. For EDT-256, multi-thread MEG decreases the evaluation time by 13.5% compared to Pipelined, by 12.6% compared to Non-Pipelined, and is very close to the lowest time achieved by Pipelined+ (with only 1.5% overhead). This indicates multi-thread MEG can effectively reduce the execution time.

On the contrary, the execution time of single-thread MEG increases significantly when the burst size becomes larger. When the burst size increases from 1 MB to 2 MB and beyond, the execution time of single-thread MEG increases by 21.6% to 31.5% for AES-128, and by 3.8% to 8.4% for EDT-256. As a result, single-thread MEG takes longer time than multi-thread MEG, with increments of 21.9% to 31.9% for AES-128, and 3.9% to 8.3% for EDT-256. Therefore multi-thread MEG has advantages in execution time compared to single-thread MEG.



(a) AES-128 (b) EDT-256

Figure 5.5: Execution time of MEG with different burst sizes, in comparison with other methods.



(a) AES-128 (b) EDT-256

Figure 5.6: Empirical cumulative distribution function (CDF) of power consumption during circuit evaluation. The burst size for MEG is 2 MB.

5.4.5 Discussion

5.4.5.1 Reason for energy reduction by MEG

The advantage of MEG in energy efficiency is because it allows the CPU and network interface to enter power saving mode when each has low workload. Figure 5.6 present the cumulative distribution function (CDF) of power consumption values during the circuit evaluation for AES-128 and EDT-256, respectively. We observe that compared to both Pipelined and Pipelined+, MEG reduces the time spent in high power state while takes similar time in low power state, thus it reduces the total energy consumption.

5.4.5.2 Reason for execution time different between single-thread MEG and multi-thread MEG

We also observe that multi-thread MEG reduces the total elapsed time than single-thread MEG when the burst size is above 2 MB. The reason is illustrated in Figure 5.2, which shows the power traces of using single-thread and multi-thread MEG to evaluate EDT-256 using 16 MB burst size. For single-thread MEG, there are periodical power consumption decreasing periods. These decreases occur at the same time when the evaluation of currently received gates is finished, and the evaluator is waiting for the new batch of gates to be completely transmitted. In contrast, there are no such power decreases in the multi-thread MEG traces, because the gates evaluation thread does not need to pause and wait for the gates receiving thread to finish, which reduces the circuit evaluation time compared with the single-thread MEG.

5.5 Conclusion and Future Work

In this work, we present a memory- and energy-efficient garbled circuit evaluation mechanism (MEG) on smartphones. The main techniques used by MEG are batch data transmission, multi-threading execution, and slow-start strategy. Our experiment results show that MEG combines the advantages of non-pipelined method for energy efficiency and pipelined method for memory saving. Compared to the non-pipelined method, MEG reduces memory consumption by 97.5% for EDT-256 when the batch size is 2 MB. For energy consumption, compared to the pipelined method, MEG brings reductions of up to 42% for AES-128 and up to 23% for EDT-256. MEG only has a minimal energy overhead (1% to 8%) compared to the non-pipelined method. Compared to both the pipelined and non-pipelined methods, the multi-threading implementation of MEG significantly reduces the circuit evaluation time on smartphones.

Chapter 6

Conclusion

In this dissertation, we propose solutions to enhance energy efficiency and privacy protection on smart devices.

First, we demonstrated that it is possible to accurately infer browsing activity on a smartphone using USB power consumption measurements. Our work is the first to study this side-channel attack on smartphones, and to analyze a multitude of factors that affect the traces that are collected during the attack, such as: battery charging level, user interaction with the touchscreen, trace length, time between collection of training and testing traces, WiFi and LTE connectivity, training and testing device mismatch, and website characteristics such as type of connection (HTTP or HTTPS) and location of the host serving the webpage relative to the smartphone. We performed extensive experiments to validate our approach. Our results show that the attack successfully identifies webpages loaded using the standard Android mobile browser at least 91.7% of the times within four seconds for the automated dataset. Overall, our results show that the attack is highly effective, because webpage loading generates power signatures that are distinctive and consistent.

Second, we demonstrated a technique based on USB power analysis that allows a malicious charging station to identify which webpages are loaded on a smartphone using Tor. To our knowledge, this is the first work to study attacks on Tor based on smartphone power side-channels. We validated our attack under realistic smartphone constraints

by collecting and analyzing power traces under several scenarios, including different networks, different devices, and different battery charging levels. We correctly identified webpages visited using the official mobile Tor browser. We achieved accuracies between 36.58% and 85.7% when the battery was fully charged, and between 34.5% and 46% when the battery level was at 30%-50%.

Third, we proposed memory and energy efficient garbled circuit evaluation (MEG) on smartphones. We evaluated MEG using different burst size and compared it with existing circuit evaluation methods. The evaluation results show that the energy consumption of MEG is significantly lower than the current pipelining schemes and close to that of the no-pipelining method. MEG also reduces the evaluation time significantly compared to the basic pipelining. We also evaluated two versions of MEG: single-threaded, and multi-threaded. We found that the energy consumption of multi-thread and single-thread MEG is close, but the execution time using multi-thread MEG is shorter and more consistent than using single-thread MEG.

As future work, we would like to study multitude of factors that we did not address in this paper, such as number of applications installed on the smartphone, background processes, network congestion, WiFi/LTE signal strength, and specific user interaction. We will also include a large webpage dataset to fully characterize how our technique extend to the entire web. Further, we will investigate how to improve the identification accuracy for the scenario when battery charging level is low. For the work of MEG, we will investigate the performance of MEG on more garbled circuits, smartphones, and network types. We will also create a model to select the best burst size based on factors such as the circuit size, the run-time status of the smartphone (such as CPU load and available RAM quantity), as well as network condition.

Bibliography

[1] Cell phone battery statistics across major us cities. https://veloxity.us/phone-battery-statistics/. Accessed: 2015-09-07.

[2] Power up: A guide to US airport charging stations – Cheapflights. http://www.cheapflights.com/news/power-up-a-guide-to-us-airport-charging-stations/#ewr. Accessed: 2016-04-04.

[3] Briant Park Blog: Solar-powered charging stations land in Bryant Park. http://blog.bryantpark.org/2014/07/solar-powered-charging-stations-land-in.html. Accessed: 2016-04-04.

[4] Solar-powered phone charging stations launch in union square. https://www.dnainfo.com/new-york/20130620/union-square/solar-powered-phone-charging-stations-launch-union-square. Accessed: 2016-04-04.

[5] Chargeport hotel charging station. http://www.teleadapt.com/hospitality-products/power-charging/chargeport. Accessed: 2016-04-04.

[6] Behind the charge: A big challenge for hospitals. http://www.mkelements.com/blog/behind-charge-big-challenge-hospitals. Accessed: 2016-04-04.

[7] Beware of juice-jacking. http://krebsonsecurity.com/2011/08/beware-of-juice-jacking/. Accessed: 2016-04-04.

[8] SyncStop: Charge your mobile phone safely. http://syncstop.com. Accessed: 2016-04-04.

[9] Tor project: Anonymity online. https://www.torproject.org/, 2017. Accessed: 2017-03-10.

[10] Roger Dingledine, Nick Mathewson, and Paul Syverson. Tor: The second-generation onion router. In *Proceedings of the 13th Conference on USENIX Security Symposium - Volume 13*, SSYM'04, pages 21–21, Berkeley, CA, USA, 2004. USENIX Association.

[11] Tor metrics. https://metrics.torproject.org, 2017. Accessed: 2017-03-10.

[12] Tor on android. https://www.torproject.org/docs/android.html.en, 2017. Accessed: 2017-03-10.

[13] ShaneS. Clark, Hossen Mustafa, Benjamin Ransford, Jacob Sorber, Kevin Fu, and Wenyuan Xu. Current events: Identifying webpages by tapping the electrical outlet. In *Computer Security - ESORICS 2013*, Jason Crampton, Sushil Jajodia, and Keith Mayes, editors, volume 8134 of *Lecture Notes in Computer Science*, pages 700–717. Springer Berlin Heidelberg, 2013.

[14] Aaron Carroll and Gernot Heiser. An analysis of power consumption in a smartphone. In *Proceedings of the 2010 USENIX Conference on USENIX Annual Technical Conference*, USENIXATC'10, pages 21–21, Berkeley, CA, USA, 2010. USENIX Association.

[15] What you think you know about the web is wrong. http://time.com/12933/what-you-think-you-know-about-the-web-is-wrong/. Accessed: 2016-04-04.

[16] Daniel Genkin, Itamar Pipman, and Eran Tromer. Get your hands off my laptop: Physical side-channel key-extraction attacks on pcs. In *Cryptographic Hardware and Embedded Systems - CHES 2014*, Lejla Batina and Matthew Robshaw, editors, volume 8731 of *Lecture Notes in Computer Science*, pages 242–260. Springer Berlin Heidelberg, 2014.

[17] Yan Michalevsky, Aaron Schulman, Gunaa Arumugam Veerapandian, Dan Boneh, and Gabi Nakibly. Powerspy: Location tracking using mobile device power analysis. In *24th USENIX Security Symposium (USENIX Security 15)*, pages 785–800, Washington, D.C., August 2015. USENIX Association.

[18] Lin Yan, Yao Guo, Xiangqun Chen, and Hong Mei. A study on power side channels on mobile devices. In *The Seventh Asia-Pacific Symposium on Internetware*, Internetware'15, 2015.

[19] Andrew Hintz. Fingerprinting websites using traffic analysis. In *Proceedings of the 2Nd International Conference on Privacy Enhancing Technologies*, PET'02, pages 171–178, Berlin, Heidelberg, 2003. Springer-Verlag.

[20] Liming Lu, Ee-Chien Chang, and Mun Choon Chan. Website fingerprinting and identification using ordered feature sequences. In *Proceedings of the 15th European Conference on Research in Computer Security*, ESORICS'10, pages 199–214, Berlin, Heidelberg, 2010. Springer-Verlag.

[21] Shuo Chen, Rui Wang, XiaoFeng Wang, and Kehuan Zhang. Side-channel leaks in web applications: A reality today, a challenge tomorrow. In *Proceedings of the 2010 IEEE Symposium on Security and Privacy*, SP '10, pages 191–206, Washington, DC, USA, 2010. IEEE Computer Society.

[22] Badusb - on accessories that turn evil. https://pacsec.jp/psj14/PSJ2014_Karsten_Nohl_141112.BadUSB-Pacsec.KN01.pdf. Accessed: 2015-09-07.

[23] Usb attacks need physical access right? not any more... https://www.blackhat.com/docs/asia-14/materials/Davis/Asia-14-Davis-USB-Attacks-Need-Physical-Access-Right-Not-Any-More.pdf. Accessed: 2015-09-07.

[24] Dominik Herrmann, Rolf Wendolsky, and Hannes Federrath. Website fingerprinting: Attacking popular privacy enhancing technologies with the multinomial naïve-bayes

classifier. In *Proceedings of the 2009 ACM Workshop on Cloud Computing Security*, CCSW '09, pages 31–42, New York, NY, USA, 2009. ACM.

[25] Andriy Panchenko, Lukas Niessen, Andreas Zinnen, and Thomas Engel. Website fingerprinting in onion routing based anonymization networks. In *Proceedings of the 10th Annual ACM Workshop on Privacy in the Electronic Society*, WPES '11, pages 103–114, New York, NY, USA, 2011. ACM.

[26] Xiang Cai, Xin Cheng Zhang, Brijesh Joshi, and Rob Johnson. Touching from a distance: Website fingerprinting attacks and defenses. In *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, CCS '12, pages 605–616, New York, NY, USA, 2012. ACM.

[27] Timothy G. Abbott, Katherine J. Lai, Michael R. Lieberman, and Eric C. Price. Browser-based attacks on tor. In *Proceedings of the 7th International Conference on Privacy Enhancing Technologies*, PET'07, pages 184–199, Berlin, Heidelberg, 2007. Springer-Verlag.

[28] Brian N. Levine, Michael K. Reiter, Chenxi Wang, and Matthew Wright. *Timing Attacks in Low-Latency Mix Systems*, pages 251–265. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004.

[29] Nicholas Hopper, Eugene Y. Vasserman, and Eric Chan-Tin. How much anonymity does network latency leak? In *Proceedings of the 14th ACM Conference on Computer and Communications Security*, CCS '07, pages 82–91, New York, NY, USA, 2007. ACM.

[30] Yixin Sun, Anne Edmundson, Laurent Vanbever, Oscar Li, Jennifer Rexford, Mung Chiang, and Prateek Mittal. Raptor: Routing attacks on privacy in tor. In *Proceedings of the 24th USENIX Conference on Security Symposium*, SEC'15, pages 271–286, Berkeley, CA, USA, 2015. USENIX Association.

[31] Kevin Bauer, Damon McCoy, Dirk Grunwald, Tadayoshi Kohno, and Douglas Sicker. Low-resource routing attacks against tor. In *Proceedings of the 2007 ACM Workshop on Privacy in Electronic Society*, WPES '07, pages 11–20, New York, NY, USA, 2007. ACM.

[32] Albert Kwon, Mashael AlSabah, David Lazar, Marc Dacier, and Srinivas Devadas. Circuit fingerprinting attacks: Passive deanonymization of tor hidden services. In *24th USENIX Security Symposium (USENIX Security 15)*, pages 287–302, Washington, D.C., 2015. USENIX Association.

[33] Steven J. Murdoch and George Danezis. Low-cost traffic analysis of tor. In *Proceedings of the 2005 IEEE Symposium on Security and Privacy*, SP '05, pages 183–195, Washington, DC, USA, 2005. IEEE Computer Society.

[34] Sambuddho Chakravarty, Angelos Stavrou, and Angelos D. Keromytis. Traffic analysis against low-latency anonymity networks using available bandwidth estimation. In *Proceedings of the 15th European Conference on Research in Computer Security*, ESORICS'10, pages 249–267, Berlin, Heidelberg, 2010. Springer-Verlag.

[35] X. Wang, S. Chen, and S. Jajodia. Network flow watermarking attack on low-latency anonymous communication systems. In *2007 IEEE Symposium on Security and Privacy (SP '07)*, pages 116–130, May 2007.

[36] Marco Valerio Barbera, Vasileios P. Kemerlis, Vasilis Pappas, and Angelos D. Keromytis. *CellFlood: Attacking Tor Onion Routers on the Cheap*, pages 664–681. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[37] Q. Yang, P. Gasti, G. Zhou, A. Farajidavar, and K. S. Balagani. On inferring browsing activity on smartphones via usb power analysis side-channel. *IEEE Transactions on Information Forensics and Security*, 12(5):1056–1066, May 2017.

[38] Riccardo Spolaor, Laila Abudahi, Veelasha Moonsamy, Mauro Conti, and Radha

Poovendran. No free charge theorem: a covert channel via USB charging cable on mobile devices. *CoRR*, abs/1609.02750, 2016.

[39] Vipul Goyal, Payman Mohassel, and Adam Smith. Efficient two party and multi party computation against covert adversaries. In *Proceedings of the Theory and Applications of Cryptographic Techniques 27th Annual International Conference on Advances in Cryptology*, EUROCRYPT'08, pages 289–306, Berlin, Heidelberg, 2008. Springer-Verlag.

[40] Vladimir Kolesnikov and Thomas Schneider. Improved garbled circuit: Free xor gates and applications. *Automata, Languages and Programming*, pages 486–498, 2008.

[41] Benny Pinkas, Thomas Schneider, Nigel P Smart, and Stephen C Williams. Secure two-party computation is practical. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 250–267. Springer, 2009.

[42] Yan Huang, David Evans, Jonathan Katz, and Lior Malka. Faster secure two-party computation using garbled circuits. In *USENIX Security Symposium*, volume 201, 2011.

[43] Benjamin Kreuter, Abhi Shelat, and Chih-Hao Shen. Billion-gate secure computation with malicious adversaries. In *USENIX Security Symposium*, volume 12, pages 285–300, 2012.

[44] Yehuda Lindell and Benny Pinkas. Secure two-party computation via cut-and-choose oblivious transfer. *Journal of cryptology*, 25(4):680–722, 2012.

[45] Yan Huang, Peter Chapman, and David Evans. Privacy-preserving applications on smartphones. In *HotSec*, 2011.

[46] Benjamin Mood, Lara Letaw, and Kevin Butler. Memory-efficient garbled circuit generation for mobile devices. In *International Conference on Financial Cryptography and Data Security*, pages 254–268. Springer, 2012.

[47] Jaroslav Šeděnka, Sathya Govindarajan, Paolo Gasti, and Kiran S Balagani. Secure outsourced biometric authentication with performance evaluation on smartphones. *IEEE Transactions on Information Forensics and Security*, 10(2):384–396, 2015.

[48] Henry Carter, Benjamin Mood, Patrick Traynor, and Kevin Butler. Secure outsourced garbled circuit evaluation for mobile devices. *Journal of Computer Security*, 24(2):137–180, 2016.

[49] Paolo Gasti, Jaroslav Šeděnka, Qing Yang, Gang Zhou, and Kiran S Balagani. Secure, fast, and energy-efficient outsourced authentication for smartphones. *IEEE Transactions on Information Forensics and Security*, 11(11):2556–2571, 2016.

[50] All about skimmers – krebs on security. http://krebsonsecurity.com/all-about-skimmers/. Accessed: 2016-03-02.

[51] Alexa global website traffic ranking. http://www.alexa.com/topsites/global. Accessed: 2015-09-07.

[52] Max77818: Dual input, power path, 3A switching mode charger with FG. http://datasheets.maximintegrated.com/en/ds/MAX77818.pdf. Accessed: 2015-09-07.

[53] Charging lithium-ion batteries: Not all charging systems are created equal. https://www.microchip.com/stellent/groups/designcenter_sg/documents/market_communication/en028061.pdf. Accessed: 2015-09-07.

[54] Triple output power supply Agilent model E3630A. http://cp.literature.agilent.com/litweb/pdf/5959-5329.pdf. Accessed: 2015-09-07.

[55] NI USB-6211 DAQ. http://sine.ni.com/nips/cds/view/p/lang/en/nid/203224. Accessed: 2015-09-07.

[56] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[57] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Mach. Learn.*, 20(3):273–297, September 1995.

[58] John C. Platt. Advances in kernel methods. In *Advances in Kernel Methods*, Bernhard Schölkopf, Christopher J. C. Burges, and Alexander J. Smola, editors, chapter Fast Training of Support Vector Machines Using Sequential Minimal Optimization, pages 185–208. MIT Press, Cambridge, MA, USA, 1999.

[59] Eamonn J. Keogh and Michael J. Pazzani. Scaling up dynamic time warping for datamining applications. In *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '00, pages 285–289, New York, NY, USA, 2000. ACM.

[60] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. The weka data mining software: An update. *SIGKDD Explor. Newsl.*, 11(1):10–18, November 2009.

[61] P. Duhamel and M. Vetterli. Fast fourier transforms: A tutorial review and a state of the art. *Signal Process.*, 19(4):259–299, April 1990.

[62] Average mobile page load time for a fortune 100 company is about 5 seconds. `http://cl.ly/2v0g2B1c0R00`. Accessed: 2016-02-09.

[63] Sebastian Kaune, Konstantin Pussep, Christof Leng, Aleksandra Kovacevic, Gareth Tyson, and Ralf Steinmetz. Modelling the internet delay space based on geographical locations. In *Parallel, Distributed and Network-based Processing, 2009 17th Euromicro International Conference on*, pages 301–310. IEEE, 2009.

[64] Arin (american registry for internet numbers) whois tool. `https://whois.arin.net`. Accessed: 2015-12-12.

[65] Apnic (asia pacific network information centre) whois tool. `https://wq.apnic.net/apnic-bin/whois.pl`. Accessed: 2015-12-12.

[66] A quick, simple guide to tor and the internet of things (so far). https://blog.torproject.org/blog/quick-simple-guide-tor-and-internet-things-so-far, 2016. Accessed: 2017-05-12.

[67] Home assistant: Tor onion service configuration. https://home-assistant.io/docs/ecosystem/tor/, 2017. Accessed: 2017-05-12.

[68] Tor: Hidden service protocol. https://www.torproject.org/docs/hidden-services, 2017. Accessed: 2017-03-10.

[69] Orfox: A tor browser for android. https://guardianproject.info/apps/orfox/, 2017. Accessed: 2017-03-10.

[70] Dp800 series dc power supplies. https://www.rigolna.com/products/dc-power-supplies/dp800/. Accessed: 2017-05-25.

[71] Qualcomm quick charge 2.0 chipset. https://www.qualcomm.com/products/quick-charge-2, 2017. Accessed: 2017-03-10.

[72] Battery charging specification revision 1.2. http://www.usb.org/developers/docs/devclass_docs/. Accessed: 2015-09-07.

[73] Usb-6211 (multifunction i/o device). http://www.ni.com/en-us/support/model.usb-6211.html. Accessed: 2017-05-25.

[74] Https everywhere. https://www.eff.org/https-everywhere, 2017. Accessed: 2017-03-10.

[75] The hidden wiki. http://zqktlwi4fecvo6ri.onion/wiki/index.php/Main_Page, 2017. Accessed: 2017-03-10.

[76] Tor path specification. https://gitweb.torproject.org/torspec.git/tree/path-spec.txt, 2015. Accessed: 2017-03-10.

[77] S. Davis and P. Mermelstein. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 28(4):357–366, Aug 1980.

[78] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. The weka data mining software: An update. *SIGKDD Explor. Newsl.*, 11(1):10–18, November 2009.

[79] John C. Platt. Advances in kernel methods. chapter Fast Training of Support Vector Machines Using Sequential Minimal Optimization, pages 185–208. MIT Press, Cambridge, MA, USA, 1999.

[80] Andrew C. Yao. Protocols for secure computations. In *Proceedings of the 23rd Annual Symposium on Foundations of Computer Science*, SFCS '82, pages 160–164, Washington, DC, USA, 1982. IEEE Computer Society.

[81] Yehuda Lindell and Benny Pinkas. Secure multiparty computation for privacy-preserving data mining. *IACR Cryptology ePrint Archive*, 2008:197, 2008.

[82] Marina Blanton and Paolo Gasti. *Secure and Efficient Protocols for Iris and Fingerprint Identification*, pages 190–209. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.

[83] Sriram Nandha Premnath and Zygmunt J. Haas. Supporting privacy of computations in mobile big data systems. *Future Internet*, 8(2), 2016.

[84] G. Zhuo, Q. Jia, L. Guo, M. Li, and P. Li. Privacy-preserving verifiable set operation in big data for cloud-assisted mobile crowdsourcing. *IEEE Internet of Things Journal*, 4(2):572–582, April 2017.

[85] Andrew Chi-Chih Yao. How to generate and exchange secrets. In *Proceedings of the 27th Annual Symposium on Foundations of Computer Science*, SFCS '86, pages 162–167, Washington, DC, USA, 1986. IEEE Computer Society.

[86] Fahad R. Dogar, Peter Steenkiste, and Konstantina Papagiannaki. Catnap: Exploiting high bandwidth wireless interfaces to save energy for mobile devices. In *Proceedings of the 8th International Conference on Mobile Systems, Applications, and Services*, MobiSys '10, pages 107–122, New York, NY, USA, 2010. ACM.

[87] Shuo Deng and Hari Balakrishnan. Traffic-aware techniques to reduce 3g/lte wireless energy consumption. In *Proceedings of the 8th International Conference on Emerging Networking Experiments and Technologies*, CoNEXT '12, pages 181–192, New York, NY, USA, 2012. ACM.

[88] Michael O. Rabin. How to exchange secrets with oblivious transfer, 2005. Harvard University Technical Report 81 talr@watson.ibm.com 12955 received 21 Jun 2005.

[89] Dahlia Malkhi, Noam Nisan, Benny Pinkas, and Yaron Sella. Fairplay—a secure two-party computation system. In *Proceedings of the 13th Conference on USENIX Security Symposium - Volume 13*, SSYM'04, pages 20–20, Berkeley, CA, USA, 2004. USENIX Association.

[90] Wilko Henecka, Stefan K ögl, Ahmad-Reza Sadeghi, Thomas Schneider, and Immo Wehrenberg. Tasty: Tool for automating secure two-party computations. In *Proceedings of the 17th ACM Conference on Computer and Communications Security*, CCS '10, pages 451–462, New York, NY, USA, 2010. ACM.

[91] Kimmo Järvinen, Vladimir Kolesnikov, Ahmad-Reza Sadeghi, and Thomas Schneider. *Garbled Circuits for Leakage-Resilience: Hardware Implementation and Evaluation of One-Time Programs*, pages 383–397. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

[92] Lior Malka. Vmcrypt: Modular software architecture for scalable secure computation. In *Proceedings of the 18th ACM Conference on Computer and Communications Security*, CCS '11, pages 715–724, New York, NY, USA, 2011. ACM.

[93] Yifan Zhang, Xudong Wang, Xuanzhe Liu, Yunxin Liu, Li Zhuang, and Feng Zhao. Towards better cpu power management on multicore smartphones. In *Proceedings of the Workshop on Power-Aware Computing and Systems*, HotPower '13, pages 11:1–11:5, New York, NY, USA, 2013. ACM.

[94] Monsoon power monitor. http://www.msoon.com/LabEquipment/PowerMonitor/. Accessed: 2017-10-9.